IBM DB2 Universal Database

Kom godt i gang med SQL

Version 7

S511-5803-00

IBM DB2 Universal Database

Kom godt i gang med SQL

Version 7

S511-5803-00

Læs de generelle oplysninger under "Tillæg C. Om dette dokument" på side 111, før oplysningerne i denne bog og det tilhørende program anvendes.

Oversat af IBM Sprogcenter.

Indholdsfortegnelse

Indledning

Denne bog er beregnet som en introduktion til SQL (Structured Query Language) og relationsdatabaser. Bogen:

- Beskriver grundlæggende begreber i SQL, der bruges i DB2-programmet.
- Forklarer, hvordan du arbejder med databasen.
- Indeholder enkle eksempler på opgaver.

Inden du udfører eksemplerne i bogen skal du, hvis du er administrator:

- Installere og konfigurere serveren som beskrevet i brugervejledningen (*Quick Beginnings*) til det styresystem, du bruger. Oprette SAMPLE-databasen vha. funktionen Første trin. SAMPLE-databasen kan også oprettes fra kommandolinien. Der er flere oplysninger i *SQL Reference*. Bemærk: Undlad at placere egne data i DB2 SAMPLE-databasen.
- Oprette bruger-id'en til DB2-administratoren som beskrevet i brugervejledningen (*Quick Beginnings*).

Hvis du ikke er systemadministrator, skal du sørge for at have en gyldig bruger-id og de relevante autorisationer og rettigheder, så du kan få adgang til SAMPLE-databasen.

Se også...

Følgende bøger indeholder flere oplysninger om specifikke emner:

Brugervejledning (*Quick Beginnings*)	Installation og brug af databasesystemet.
SQL Reference	SQL-meddelelser.
Administration Guide	Design, implementering og vedligeholdelse af en lokal database eller en database i et client/server-miljø.
Application Development Guide	Applikationsudvikling samt programmering, kompilering og udførelse af applikationer, der anvender indlejret SQL til at få adgang til databasen, eller som køres som lagrede DB2-procedurer vha. SQL-proceduresproget (eller et andet understøttet programmeringssprog).

Typografi

Sådan fremhæves tekst i bogen:

Fed	I eksemplerne angiver fed skrift kommandoer og nøgleord, som er defineret af systemet.

Kursiv	Kursiveret skrift kan angive følgende: • Et nyt udtryk • En henvisning til en anden informationskilde.
STORE BOGSTAVER	Store bogstaver kan angive følgende: • Kommandoer og nøgleord, som er defineret af systemet • Eksempler på specifikke dataværdier eller kolonnenavne.

Kapitel 1. Relationsdatabaser og SQL

I en *relationsdatabase* gemmes data i *tabeller*. En tabel er en samling *rækker* og *kolonner*. I figur 1 på side 3 er der et grafisk eksempel på en tabel. Kolonner (lodret) og rækker (vandret) er markeret i figuren. *SQL (Structured Query Language)* bruges til at hente eller opdatere data med angivelse af kolonner, tabeller og de forskellige koblinger mellem dem.

SQL er et standardiseret sprog til definition og manipulation af data i en relationsdatabase. SQL-sætninger udføres af et *databasesystem*. Et databasesystem er et computerprogram, som styrer data.

En *inddelt* relationsdatabase er en relationsdatabase, hvor data styres på tværs af flere afsnit (også kaldt *noder*). En enkel måde at opfatte afsnit på er at betragte hvert afsnit som en fysisk computer. I denne bog er hovedvægten lagt på databaser med et enkelt afsnit.

Du kan få adgang til SAMPLE-databasen og udføre eksemplerne i bogen vha. *interaktiv* SQL ved at bruge en grænseflade som f.eks. en DB2-kommandolinie eller kommandocentralen.

Kapitel 2. Organisering af data

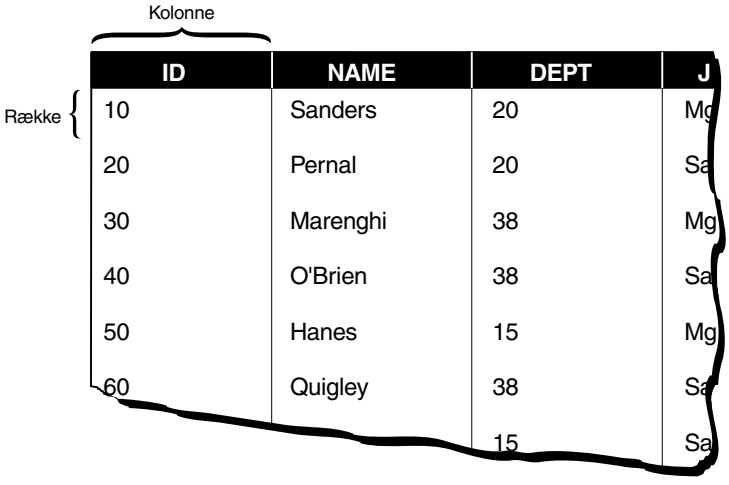
I dette kapitel beskrives vigtige begreber i forbindelse med *tabeller, udpluk* og *skemaer*. Det er en generel oversigt over de forskellige byggeklodser i en relationsdatabase. Det sidste afsnit indeholder en kort beskrivelse af nogle af de vigtigste og mest udbredte datatyper.

Tabeller

Tabeller er logiske strukturer, som består af et fast antal *kolonner* og et variabelt antal *rækker*. En kolonne er et sæt værdier af samme datatype. En række er et antal værdier, der udgør en enkelt record i tabellen. Rækkerne i tabellen er ikke nødvendigvis sorteret. Hvis du vil sortere resultatrækker, skal du eksplicit angive det i den SQL-sætning, du bruger til at hente data i tabellen. Dér hvor en kolonne og en række skærer hinanden, findes et dataelement, som kaldes en *værdi*. I figur 1 er f.eks. 'Sanders' en værdi.

En *basistabel* indeholder brugerdata. Den oprettes ved hjælp af sætningen CREATE TABLE. En *resultattabel* er et sæt rækker, som databasesystemet vælger eller genererer fra én eller flere basistabeller som svar på en forespørgsel.

I figur 1 vises en del af en tabel. Kolonner og rækker er angivet.



Figur 1. Tabel

Udpluk

Et *udpluk* (view) er en anden måde at få vist data i én eller flere tabeller. Det er et dynamisk vindue med tabeloplysninger.

Med udpluk kan flere brugere få vist de samme data på forskellige måder. Man kunne forestille sig, at flere brugere skal have adgang til en tabel med medarbejderdata. En leder ser data om sine medarbejdere men ikke om medarbejdere i en anden afdeling. En personalemedarbejder ser alle medarbejderes ansættelsesdato, men ikke deres løn, mens en regnskabsmedarbejder ser lønoplysninger, men ikke ansættelsesdatoer. Begge brugere benytter et udpluk af den egentlige tabel. Udplukkene vises som tabeller med deres eget navn.

En af fordelene ved udpluk er, at det er muligt af begrænse adgangen til fortrolige data. Forskellige personer kan have adgang til forskellige datakolonner eller -rækker.

Skemaer

Et *skema* er en samling navngivne objekter, f.eks. tabeller og udpluk. Et skema er en logisk klassificering af objekterne i databasen.

Et skema oprettes implicit, når du opretter en tabel, et udpluk eller et andet navngivet objekt. Det kan også oprettes eksplicit vha. sætningen CREATE SCHEMA.

Når du opretter et navngivet objekt, kan du *kvalificere* (tilknytte) dets navn med navnet på et skema. Navngivne objekter har todelte navne, hvor første del er navnet på det skema, som objektet er knyttet til. Hvis du ikke angiver et skemanavn, knyttes objektet til standardskemaet. Navnet på standardskemaet er *autorisations-id'en* for den bruger, der udfører sætningen.

Ved interaktiv SQL, som benyttes til eksemplerne her i bogen, er autorisations-id'en den bruger-id, der er angivet i CONNECT-sætningen. Hvis navnet på en tabel for eksempel er STAFF, og den angivne bruger-id er BRUGERID, så er det kvalificerede navn BRUGERID.STAFF. Der er flere oplysninger om CONNECT-sætningen under "Opret forbindelse til database" på side 18.

Nogle skemanavne er reserveret. Således ligger *indbyggede funktioner* i skemaet SYSIBM, mens forudinstallerede *brugerdefinerede funktioner* hører til i skemaet SYSFUN. Der er flere oplysninger om sætningen CREATE SCHEMA i *SQL Reference*.

Datatyper

Datatyper angiver gyldige værdier for konstanter, kolonner, værtsvariabler, funktioner, udtryk og specialregistre. I dette afsnit beskrives de datatype, der bruges i eksemplerne. Der er en fuldstændig oversigt over og beskrivelse af datatyper i *SQL Reference*.

Strenge af typen Character

En *Character-streng* er en bytesekvens. Længden på strengen er antallet af byte i sekvensen. Hvis længden er nul, kaldes værdien en *tom streng*.

Character-streng med fast længde

CHAR(x) er en streng med fast længde. Længdeangivelsen x skal være mellem 1 og 254.

Character-streng med variabel længde

Der er tre typer Character-strenge med variabel længde: VARCHAR, LONG VARCHAR og CLOB.

VARCHAR(x)-strenge har variabel længde, så en streng med længden 9 kan godt indsættes i VARCHAR(15), men strengens længde er stadig 9.

Under "Store objekter (LOB)" på side 67 er der flere oplysninger om CLOB.

Dobbeltbytestreng af typen Character

En *dobbeltbyte Character-streng* er en sekvens af dobbeltbyte tegn.

Dobbeltbyte Character-streng med fast længde

GRAPHIC(x) er en streng med fast længde. Længdeangivelsen x skal være mellem 1 og 127.

Dobbeltbyte Character-streng med variabel længde

Der er tre typer dobbeltbyte Character-strenge med variabel længde: VARGRAPHIC, LONG VARGRAPHIC og DBCLOB. Under "Store objekter (LOB)" på side 67 er der flere oplysninger om DBCLOB.

Binær streng

En *binær streng* er en bytesekvens. Den bruges til opbevaring af specielle data, f.eks. billeder. Et binært stort objekt (BLOB) er en binær streng. Der er flere oplysninger under "Store objekter (LOB)" på side 67.

Tal

Alle tal har et fortegn og en *præcision*. Præcisionen er antallet af bit eller cifre, ekskl. fortegnet.

SMALLINT
SMALLINT (lille heltal) er et tobyte heltal med en præcision på 5 cifre.

INTEGER
INTEGER (heltal) er et firebyte heltal med en præcision på 10 cifre.

BIGINT
BIGINT (stort heltal) er et heltal på 8 byte med en præcision på 19 cifre.

REAL *REAL (enkeltpræcisionstal med flydende decimaltegn)* er en 32-bit tilnærmelse til et reelt tal.

DOUBLE
DOUBLE (dobbeltpræcisionstal med flydende decimaltegn) er en 64-bit tilnærmelse til et reelt tal. DOUBLE kaldes også FLOAT.

DECIMAL(p,a)

DECIMAL er et decimaltal. Decimaltegnets position afgøres af tallets *præcision (p)* og *antal decimaler (a)*. Præcisionen er det samlede antal cifre, som skal være mindre end 32. Antallet af decimaler er antallet af cifre til højre for decimaltegnet, som altid er mindre end eller lig med værdien for præcision. Hvis intet andet angives, bruges standardværdierne 5 for præcision og 0 for antal decimaler.

Værdier for dato/klokkeslæt

Værdier for dato/klokkeslæt er datoer, klokkeslæt og tidsstempler (en Character-streng på 14 cifre, der repræsenterer en gyldig dato/klokkeslætværdi i formatet *ååååmmddttmmss*). Værdierne kan benyttes i visse matematiske funktioner og strengfunktioner og er kompatible med visse strenge, men de er hverken strenge eller tal.[1]

Dato En *dato* er en tredelt værdi (år, måned og dag).

Klokkeslæt
Et *klokkeslæt* er en tredelt værdi (timer, minutter og sekunder), som angiver en tid på døgnet.

1. I bogen her bruges ISO-værdierne for dato/klokkeslæt.

Tidsstempel

Et *tidsstempel* er en syvdelt værdi (år, måned, dag, timer, minutter, sekunder og mikrosekunder), der angiver en dato og et klokkeslæt.

NULL-værdi

Værdien *NULL* er en særlig værdi, som adskiller sig fra alle ikke-NULL-værdier. Den angiver, at der ikke er nogen værdi for den kolonne i rækken. NULL-værdien findes for alle datatyper.

Nedenfor vises kendetegnene for de datatyper, der benyttes i eksemplerne. Alle numeriske datatyper ligger inden for et givet interval. Intervallet vises også nedenfor. Oversigten kan bruges til at få et hurtigt overblik over datatyperne.

Datatype	Type	Kendetegn	Eksempel eller interval
CHAR(15)	Character-streng med fast længde	Maksimumslængde på 254	'Godt vejr '
VARCHAR(15)	Character-streng med variabel længde	Maksimumslængde på 32672	'Godt vejr'
SMALLINT	tal	2 byte langt præcision på 5 cifre	interval fra -32768 til 32767
INTEGER	tal	4 byte langt præcision på 10 cifre	interval fra -2147483648 til 2147483647
BIGINT	tal	8 byte langt præcision på 19 cifre	interval fra -9223372036854775808 til 9223372036854775807
REAL	tal	enkelt præcision flydende decimaltegn 32-bit tilnærmelse	interval fra -3.402E+38 til -1.175E-37 eller 1.175E-37 til -3.402E+38 eller nul
DOUBLE	tal	dobbelt præcision flydende decimaltegn 64-bit tilnærmelse	interval fra -1.79769E+308 til -2.225E-307 eller 2.225E-307 til 1.79769E+308 eller nul
DECIMAL(5,2)	tal	præcision = 5 antal decimaler = 2	interval fra -10**31+1 til 10**31-1
DATE	dato/- klokkeslæt	tredelt værdi	1991-10-27
TIME	dato/- klokkeslæt	tredelt værdi	13.30.05

Datatype	Type	Kendetegn	Eksempel eller interval
TIMESTAMP	dato/- klok-keslæt	syvdelt værdi	1991-10-27-13.30.05.000000

Der er flere oplysninger i oversigten over datatypekompatibilitet i *SQL Reference*.

Kapitel 3. Oprettelse af tabeller og udpluk

I dette kapitel beskrives, hvordan du kan oprette og ændre tabeller og udpluk i DB2 Universal Database. Relationen mellem tabeller og udpluk vises med diagrammer og eksempler.

Der er følgende afsnit i kapitlet:

- Opret tabel og Opret udpluk
- Indsæt data
- Revidér data
- Slet data
- Revidér data vha. udpluk

Opret tabel

Ved hjælp af CREATE TABLE-sætningen kan du oprette tabeller og angive kolonnenavne og -typer samt *betingelser*. Betingelser beskrives under "Anvend betingelser og triggere til forretningsregler" på side 51.

I følgende eksempel oprettes tabellen PERS, som ligner tabellen STAFF, men har en ekstra kolonne til fødselsdato.

```
CREATE TABLE PERS
    ( ID              SMALLINT        NOT NULL,
      NAME            VARCHAR(9),
      DEPT            SMALLINT WITH DEFAULT 10,
      JOB             CHAR(5),
      YEARS           SMALLINT,
      SALARY          DECIMAL(7,2),
      COMM            DECIMAL(7,2),
      BIRTH_DATE      DATE)
```

Med denne sætning oprettes en tabel uden data. I næste afsnit beskrives, hvordan du indsætter data i en tabel.

Som eksemplet viser, skal du angive både navn og datatype for hver kolonne. Datatyper beskrives under "Datatyper" på side 5. NOT NULL er valgfrit og kan angives, hvis kolonnen ikke skal kunne indeholde NULL-værdier. Det er også valgfrit at angive standardværdier.

I CREATE TABLE-sætningen kan du også angive en række andre valg, f.eks. *entydige betingelser og referencebetingelser*. Der er flere oplysninger om disse muligheder i *SQL Reference*.

Indsæt data

En nyoprettet tabel indeholder ingen data. Hvis du vil indsætte rækker i en tabel, skal du bruge INSERT-sætningen. Den har to generelle formater:

- I det ene format bruges et VALUES-udtryk til at angive værdier for kolonnerne i én eller flere rækker. I de næste tre eksempler bruges dette generelle format.
- I det andet format angiver du ingen værdier. I stedet bruger du en *fullselect* til at identificere kolonner fra rækker i andre tabeller og/eller udpluk.

En fullselect er en SELECT-sætning, der bruges i INSERT- og CREATE VIEW-sætningerne samt efter et prædikat. En fullselect i parentes kaldes gerne en *underforespørgsel*.

Afhængigt af de standardværdier, du har valgt til tabellen, skal du for hver række, der indsættes, enten angive en værdi for hver kolonne eller acceptere standardværdien. Der er flere oplysninger om standardværdier for de forskellige datatyper i *SQL Reference*.

I dette eksempel bruges et VALUES-udtryk til at indsætte en datarække i tabellen PERS:

```
INSERT INTO PERS
   VALUES (12, 'Harris', 20, 'Sales', 5, 18000, 1000, '1950-1-1')
```

I nedenstående eksempel bruges VALUES-udtrykket til at indsætte tre rækker i tabellen PERS. Kun værdierne for id, navn og job kendes. Hvis en kolonne ikke kan indeholde NULL, og den ikke har en standardværdi, skal du angive en værdi.

Udtrykket NOT NULL i en kolonnedefinition i CREATE TABLE-sætningen kan udvides med ordene WITH DEFAULT. Hvis en kolonne er defineret som NOT NULL WITH DEFAULT eller en konstant standardværdi, som f.eks. WITH DEFAULT 10, og du ikke angiver kolonnen i listen over kolonner, bruges kolonnens standardværdi i den indsatte række. I eksemplet her er der i CREATE TABLE-sætningen kun angivet en standardværdi for kolonnen DEPT, og den er 10. Derfor indsættes afdelingsnummeret (DEPT) 10, mens alle andre kolonner, der ikke eksplicit angives en værdi for, får en NULL-værdi.

```
INSERT INTO PERS (NAME, JOB, ID)
   VALUES ('Swagerman', 'Prgmr', 500),
          ('Limoges', 'Prgmr', 510),
          ('Li', 'Prgmr', 520)
```

Med denne sætning får du vist resultatet af indsættelserne:

```
SELECT *
   FROM PERS

ID     NAME      DEPT   JOB   YEARS  SALARY    COMM      BIRTH_DATE
------ --------- ------ ----- ------ --------- --------- ----------
    12 Harris        20 Sales      5  18000.00   1000.00 01/01/1950
   500 Swagerman     10 Prgmr      -         -         - -
   510 Limoges       10 Prgmr      -         -         - -
   520 Li            10 Prgmr      -         -         - -
```

I dette tilfælde vises der ikke værdier for alle kolonner. NULL-værdier angives ved en tankestreg (–). Hvis det skal kunne lade sig gøre, skal listen over kolonnenavne svare nøjagtigt til værdierne i VALUES-udtrykket både mht. rækkefølge og datatype. Hvis kolonnenavnene udelades (som i det første eksempel), skal værdierne efter VALUES stå i samme rækkefølge som kolonnerne i den tabel, de skal indsættes i, og antallet af værdier skal være lig med antallet af kolonner i tabellen.

Alle værdier skal være kompatible med datatyperne i de kolonner, de indsættes i. Hvis en kolonne kan indeholde NULL, og der ikke angives nogen værdi for kolonne, bruges NULL-værdien i den kolonne i den indsatte række.

I nedenstående eksempel indsættes NULL-værdier i YEARS, COMM og BIRTH_DATE, da der ikke er angivet værdier for disse kolonner.

```
INSERT INTO PERS (ID, NAME, JOB, DEPT, SALARY)
   VALUES (410, 'Perna', 'Sales', 20, 20000)
```

Det andet format til INSERT-sætningen er praktisk, hvis værdier fra rækker i andre tabeller skal indsættes i en tabel. Som sagt skal du ikke angive værdier, men bruge en fullselect til at identificere kolonner fra rækker i andre tabeller og/eller udpluk.

I dette eksempel hentes data om medarbejdere i afdeling 38 fra tabellen STAFF og indsættes i tabellen PERS:

```
INSERT INTO PERS (ID, NAME, DEPT, JOB, YEARS, SALARY)
   SELECT ID, NAME, DEPT, JOB, YEARS, SALARY
      FROM STAFF
      WHERE DEPT = 38
```

Når disse data er indsat, giver følgende SELECT-sætning samme resultat som fullselecten i INSERT-sætningen.

```
SELECT ID, NAME, DEPT, JOB, YEARS, SALARY
   FROM PERS
   WHERE DEPT = 38
```

Resultatet er:

```
ID     NAME      DEPT   JOB   YEARS  SALARY
------ --------- ------ ----- ------ ---------
    30 Marenghi      38 Mgr        5  17506.75
    40 O'Brien       38 Sales      6  18006.00
    60 Quigley       38 Sales      -  16808.30
   120 Naughton      38 Clerk      -  12954.75
   180 Abrahams      38 Clerk      3  12009.75
```

Revidér data

Brug UPDATE-sætningen, hvis du vil ændre data i en tabel. Med denne sætning kan du ændre værdien for én eller flere kolonner i de rækker, som opfylder søgekriterierne i WHERE-udtrykket.

I eksemplet her opdateres oplysningerne om medarbejderen med id'en 410:

```
UPDATE PERS
   SET JOB='Prgmr', SALARY = SALARY + 300
   WHERE ID = 410
```

SET-udtrykket angiver de kolonner, der skal opdateres, og de nye værdier.

WHERE-udtrykket er valgfrit og angiver, hvilke rækker der skal opdateres. Hvis WHERE-udtrykket udelades, opdaterer databasesystemet alle rækker i tabellen eller udplukket med de angivne værdier.

I dette eksempel angives tabellen (PERS) først, derefter angives en betingelse for den række, der skal opdateres. Oplysningerne om medarbejder 410 skal ændres: Medarbejderens stilling er nu Prgmr, og lønnen er steget med 300.

Du kan ændre data i flere rækker ved at specificere et WHERE-udtryk, som gælder for to eller flere rækker. I dette eksempel får alle salgsmedarbejdere en lønstigning på 15%:

```
UPDATE PERS
   SET SALARY = SALARY * 1.15
   WHERE JOB = 'Sales'
```

Slet data

Med DELETE-sætningen kan du slette datarækker fra en tabel, hvis de opfylder søgekriterierne i WHERE-udtrykket. I eksemplet her slettes den række, hvor medarbejder-id'en er 120:

```
DELETE FROM PERS
   WHERE ID = 120
```

WHERE-udtrykket er valgfrit og angiver, hvilke rækker der skal slettes. Hvis WHERE-udtrykket udelades, sletter databasesystemet alle rækker i tabellen eller udplukket.

Du kan slette flere rækker med DELETE-udtrykket. I dette eksempel slettes alle rækker, hvor medarbejderens afdelingsnummer er 20:

```
DELETE FROM PERS
   WHERE DEPT = 20
```

Når du sletter en række, fjernes hele rækken, ikke kun specifikke kolonneværdier.

Hvis du ud over tabelindholdet vil slette selve tabellen, skal du bruge DROP TABLE-sætningen, som er beskrevet i *SQL Reference*.

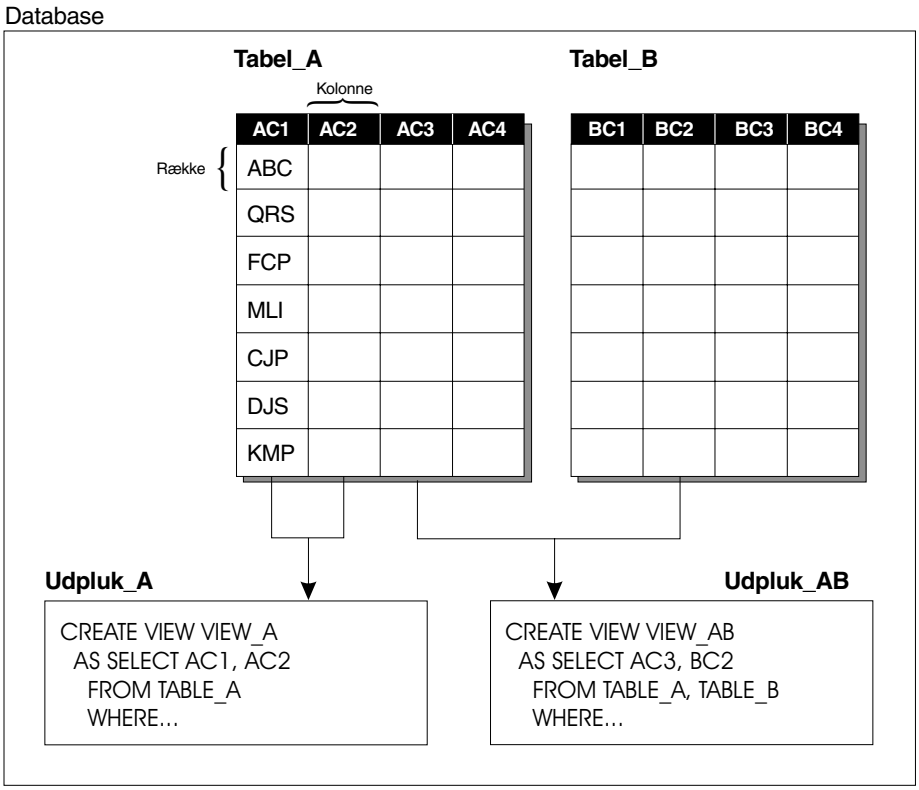
Opret udpluk

Som beskrevet under "Udpluk" på side 4 er et udpluk en anden måde at få vist data i én eller flere tabeller. Ved hjælp af udpluk kan du begrænse de oplysninger, som forskellige brugere har adgang til. Nedenfor vises relationen mellem udpluk og tabeller.

I figur 2 på side 14 begrænses adgangen i Udpluk_A til kolonnerne AC1 og AC2 i TABEL_A.

I Udpluk_AB får brugeren adgang til kolonne AC3 i TABEL_A og BC2 i TABEL_B.

Vha. Udpluk_A begrænses brugeradgangen til TABEL_A, og vha. Udpluk_AB begrænses brugeradgangen til visse kolonner i begge tabeller.



Figur 2. Relation mellem tabeller og udpluk

I nedenstående eksempel oprettes et udpluk med ikke-ledere i afdeling 20 i tabellen STAFF. Lønnen og provisionen, som vises i basistabellen, vises ikke i udplukket.

```
CREATE VIEW STAFF_ONLY
   AS SELECT ID, NAME, DEPT, JOB, YEARS
         FROM STAFF
         WHERE JOB <> 'Mgr' AND DEPT=20
```

Når du har oprettet udplukket, kan du få det vist vha. følgende sætning:

```
SELECT *
   FROM STAFF_ONLY
```

Resultatet er:

```
ID     NAME      DEPT   JOB   YEARS
------ --------- ------ ----- ------
    20 Pernal        20 Sales      8
    80 James         20 Clerk      -
   190 Sneider       20 Clerk      8
```

I endnu et eksempel kan vi bruge tabellerne STAFF og ORG til at oprette et udpluk, der viser navnet på hver afdeling og navnet på afdelingslederen. Udplukket oprettes vha. følgende sætning:

```
CREATE VIEW DEPARTMENT_MGRS
   AS SELECT NAME, DEPTNAME
         FROM STAFF, ORG
         WHERE MANAGER = ID
```

Du kan angive yderligere betingelser for tilføjelser til og opdatering af tabeller ved at bruge udtrykket WITH CHECK OPTION, når du opretter et udpluk. Databasesystemet undersøger så, om opdateringer af eller tilføjelser til udplukket er i overensstemmelse med udpluksdefinitionen. Hvis ikke, afvises de. Hvis du udelader udtrykket, kontrolleres tilføjelser og opdateringer ikke i forhold til udpluksdefinitionen. Der er flere oplysninger om WITH CHECK OPTION i *SQL Reference*.

Revidér data vha. udpluk

Ligesom SELECT-sætningen afsendes INSERT-, DELETE- og UPDATE-sætningerne mod et udpluk, som om det var en rigtig tabel. Sætningerne bruges til at ændre data i den eller de underliggende basistabel(ler). Når du igen bruger udplukket, hentes data fra de seneste basistabeller. Hvis du ikke bruger WITH CHECK OPTION-udtrykket, er det ikke sikkert, at de data, du ændrer fra udplukket, vises, næste gang du får adgang til udplukket. Yderligere kan du risikere, at dataene ikke passer til den oprindelige udpluksdefinition.

Her opdateres udplukket FIXED_INCOME:

```
CREATE VIEW FIXED_INCOME (LNAME, DEPART, JOBTITLE, NEWSALARY)
   AS SELECT NAME, DEPT, JOB, SALARY
         FROM PERS
         WHERE JOB <> 'Sales' WITH CHECK OPTION

UPDATE FIXED_INCOME
   SET NEWSALARY = SALARY * 1.10
   WHERE LNAME = 'Li'
```

Bortset fra CHECK-funktionen svarer ovenstående opdatering af udplukket til at opdatere basistabellen PERS:

```
UPDATE PERS
   SET SALARY = SALARY * 1.10
   WHERE NAME = 'Li'
     AND JOB <> 'Sales'
```

Fordi udplukket er oprettet med angivelse af WITH CHECK OPTION for betingelsen `JOB <> 'Sales'` i CREATE VIEW FIXED_INCOME, er følgende opdatering ikke mulig, når Limoges flyttes til salgsafdelingen:

```
UPDATE FIXED_INCOME
   SET JOBTITLE = 'Sales'
   WHERE LNAME = 'Limoges'
```

Kolonner, der er defineret vha. udtryk som SALARY + COMM eller SALARY * 1.25 kan ikke opdateres. Hvis du definerer et udpluk med den slags kolonner, får ejeren ikke UPDATE-rettighed til kolonnerne. INSERT-sætninger kan ikke bruges mod udpluk med den slags kolonner, men DELETE-sætninger kan.

Forestil dig, at der i tabellen PERS ikke er nogen kolonner, der ikke kan indeholde NULL. Du kan indsætte rækker i tabellen PERS fra udplukket FIXED_INCOME, selv om det ikke indeholder ID, YEARS, COMM eller BIRTH_DATE fra den underliggende tabel PERS. For kolonner, som ikke vises i udplukket, sættes værdien til NULL eller standardværdien.

Kolonnen ID i tabellen PERS kan imidlertid ikke indeholde NULL. Hvis du prøver at indsætte en række via udplukket FIXED_INCOME, prøver systemet at indsætte NULL-værdier i alle de PERS-kolonner, som er "usynlige" i udplukket. Fordi kolonnen ID ikke vises i udplukket og ikke kan indeholde NULL-værdier, tillader systemet ikke indsættelse af rækker via udplukket.

Du kan læse om regler og begrænsninger for ændring af udpluk under CREATE VIEW i *SQL Reference*.

Kapitel 4. Dataadgang vha. SQL-sætninger

I dette kapitel beskrives, hvordan du opretter forbindelse til en database og læser data vha. SQL-sætninger.

I eksemplerne vises den sætning, der skal angives, efterfulgt (i de fleste tilfælde) af resultatet, når sætningen afsendes mod SAMPLE-databasen. Sætningerne vises med store bogstaver, men kan indtastes med en vilkårlig blanding af store og små bogstaver, undtagen hvis de står i enkelte anførselstegn (') eller dobbelte anførselstegn ('').

SAMPLE-databasen, som følger med DB2 Universal Database, indeholder flere tabeller som vist i "Tillæg A. Tabeller i SAMPLE-databasen" på side 71. Databasen kan oprettes vha. startvinduet "Første trin". Du kan også oprette SAMPLE-databasen fra kommandolinien. Der er flere oplysninger i *SQL Reference*.

Der leveres flere eksempeldatabaser med DB2 Universal Database, der viser funktionerne i datavarehuscentret og OLAP. I eksemplerne i denne bog bruges kun den generelle SAMPLE-database.

Afhængigt af databasekonfigurationen skal du muligvis kvalificere de anvendte tabelnavne, dvs. sætte et skemanavn og et punktum foran dem. I eksemplerne i denne bog bruges standardskemaet BRUGERID. Du kan med andre ord kalde tabellen ORG for BRUGERID.ORG. Databaseadministratoren kan oplyse, om det er nødvendigt.

Der er følgende afsnit i kapitlet:

- Opret forbindelse til database
- Undersøg fejl
- Vælg kolonner og Vælg række
- Sortér rækker og Fjern dubletter
- Funktionsrækkefølge
- Beregn værdier vha. udtryk
- Navngiv udtryk
- Vælg data fra flere tabeller
- Underforespørgsler
- Funktioner
- Gruppering

Opret forbindelse til database

Du skal oprette forbindelse til en database, før du kan afsende forespørgsler mod den eller ændre data vha. SQL-sætninger. CONNECT-sætningen knytter en databaseforbindelse til et brugernavn.

Skriv følgende kommando på DB2-kommandolinien, hvis du vil oprette forbindelse til SAMPLE-databasen:

```
CONNECT TO SAMPLE USER BRUGERID USING  KODEORD
```

Sørg for, at vælge en bruger-id og et kodeord, der er gyldige på serversystemet.

USER er BRUGERID, og USING er KODEORD i eksemplet.

Nedenstående meddelelse viser, at forbindelsen er oprettet:

```
Oplysninger om databaseforbindelser

        Databaseserver         = DB2/NT 7.1.0
        SQL-autorisations-id   = BRUGERID
        Lokalt databasealias   = SAMPLE
```

Når forbindelsen er oprettet, kan du ændre i databasen. Der er flere oplysninger om forbindelser under CONNECT i *SQL Reference*.

Undersøg fejl

Hvis du laver en fejl, når du indtaster eksemplerne, eller hvis der opstår en fejl under udførelse af en SQL-sætning, viser databasesystemet en fejlmeddelelse. Fejlmeddelelsen består af et meddelelsesnummer, en kort forklaring og en SQLSTATE.

SQLSTATE-fejl er fejlkoder, som er fælles for alle DB2-programmer. SQLSTATE-fejl overholder ISO/ANSI SQL92-standarden.

Hvis du f.eks. angiver en forkert bruger-id eller et forkert kodeord i CONNECT-sætningen, viser databasesystemet meddelelsesnummer SQL1403N og SQLSTATE 08004. Følgende meddelelse vises:

```
SQL1403N Det angivne brugernavn og/eller kodeord er
         forkert.     SQLSTATE=08004
```

Du kan få flere oplysninger om fejlmeddelelsen ved at skrive et spørgsmålstegn (?) efterfulgt af meddelelsesnummeret eller SQLSTATE på DB2-kommandolinien:

```
        ? SQL1403N
eller
        ? SQL1403
eller
        ? 08004
```

Den næstsidste linie i beskrivelsen af fejlen SQL1403N angiver, at SQLCODE er -1403. SQLCODE er en programspecifik fejlkode. Meddelelsesnumre, der slutter med N (Notification - besked) eller C (Critical - kritisk), angiver fejl, og deres SQLCODE er negativ. Meddelelsesnumre, der slutter med W (Warning - advarsel), angiver en advarsel, og deres SQLCODE er positiv.

Vælg kolonner

Brug SELECT-sætningen, hvis du vil vælge specifikke kolonner fra en tabel. Angiv kolonnenavnene adskilt af kommaer.

I dette eksempel vælges afdelingsnavne (DEPTNAME) og afdelingsnumre (DEPTNUMB) fra tabellen ORG i SAMPLE-databasen:

```
SELECT DEPTNAME, DEPTNUMB
   FROM ORG
```

Resultatet er:

```
DEPTNAME       DEPTNUMB
-------------- --------
Head Office          10
New England          15
Mid Atlantic         20
South Atlantic       38
Great Lakes          42
Plains               51
Pacific              66
Mountain             84
```

Med en stjerne (*) kan du vælge alle kolonner i tabellen. I det næste eksempel får du vist alle kolonner og rækker i tabellen ORG:

```
SELECT *
   FROM ORG
```

Resultatet er:

```
DEPTNUMB DEPTNAME       MANAGER DIVISION   LOCATION
-------- -------------- ------- ---------- -------------
      10 Head Office        160 Corporate  New York
      15 New England         50 Eastern    Boston
      20 Mid Atlantic        10 Eastern    Washington
      38 South Atlantic      30 Eastern    Atlanta
      42 Great Lakes        100 Midwest    Chicago
      51 Plains             140 Midwest    Dallas
      66 Pacific            270 Western    San Francisco
      84 Mountain           290 Western    Denver
```

Vælg række

Hvis du vil vælge specifikke rækker i en tabel, skal du efter SELECT-sætningen med et WHERE-udtryk angive ét eller flere kriterier, som de valgte rækker skal opfylde. Et kriterium til udvælgelse af rækker fra en tabel er et *søgekriterium*.

Et søgekriterium består af ét eller flere *prædikater*. Et prædikat angiver en betingelse, som er sand eller falsk (eller ukendt) for en række. Du kan angive betingelser i WHERE-udtrykket vha. følgende grundlæggende prædikater:

Prædikat	**Funktion**
x = y	x er lig med y
x <> y	x er forskellig fra y
x < y	x er mindre end y
x > y	x er større end y
x <= y	x er mindre end eller lig med y
x >= y	x er større end eller lig med y
IS NULL/IS NOT NULL	der testes for NULL-værdier

Når du opretter søgekriterier, skal du sørge for kun at foretage beregninger for numeriske data og kun at sammenligne data, der er kompatible. Således kan tekststrenge og tal ikke sammenlignes.

Hvis du udvælger rækker på grundlag af en tegnværdi, skal værdien stå i enkelte anførselstegn, f.eks. `WHERE JOB = 'Clerk'`, og alle tegnværdier skal indtastes nøjagtig som i databasen. Hvis dataværdien er skrevet med små bogstaver i databasen, og du bruger store bogstaver, vælges ingen rækker. Hvis du udvælger rækker på grundlag af en numerisk værdi, må værdien ikke stå i anførselstegn, f.eks. `WHERE DEPT = 20`.

I følgende eksempel udvælges kun de rækker, hvor afdelingsnummeret er 20, fra tabellen STAFF:

```
SELECT DEPT, NAME, JOB
   FROM STAFF
   WHERE DEPT = 20
```

Resultatet er:

```
 DEPT   NAME      JOB
 ------ --------- -----
     20 Sanders   Mgr
     20 Pernal    Sales
     20 James     Clerk
     20 Sneider   Clerk
```

I det næste eksempel bruges AND til at angive mere end ét kriterium. Du kan angive så mange kriterier, du vil. I eksemplet udvælges de medarbejdere i afdeling 20, som har stillingsbetegnelsen "clerk", fra tabellen STAFF:

```
SELECT DEPT, NAME, JOB
   FROM STAFF
   WHERE JOB = 'Clerk'
   AND DEPT = 20
```

Resultatet er:

```
DEPT   NAME      JOB
------ --------- -----
    20 James     Clerk
    20 Sneider   Clerk
```

En NULL-værdi optræder, når der ikke angives nogen værdi, og kolonnen ikke har en standardværdi. En værdi kan også eksplicit angives til at være NULL. Værdien kan kun optræde, hvis kolonnen er defineret til at kunne indeholde NULL-værdier. Brugen af NULL-værdier beskrives under "Opret tabel" på side 9.

Brug prædikatet IS NULL eller IS NOT NULL, hvis du vil teste for NULL-værdier.

Med følgende sætning får du vist de medarbejdere, hvis provision er ukendt:

```
SELECT ID, NAME
   FROM STAFF
   WHERE COMM IS NULL
```

Resultatet er:

```
ID     NAME
------ ---------
    10 Sanders
    30 Marenghi
    50 Hanes
```

```
100 Plotz
140 Fraye
160 Molinare
210 Lu
240 Daniels
260 Jones
270 Lea
290 Quill
```

Værdien nul er ikke det samme som NULL-værdien. Med følgende sætning får du vist alle i tabellen, hvis provision er nul:

```
SELECT ID, NAME
   FROM STAFF
   WHERE COMM = 0
```

Da der ikke er nogen nuller i kolonnen COMM i eksempeltabellen, returneres en tom resultatrække.

I det næste eksempel udvælges alle rækker, hvor værdien for YEARS i tabellen STAFF er højere end 9:

```
SELECT NAME, SALARY, YEARS
   FROM STAFF
   WHERE YEARS > 9
```

Resultatet er:

```
NAME      SALARY    YEARS
--------- --------- ------
Hanes      20659.80     10
Lu         20010.00     10
Jones      21234.00     12
Quill      19818.00     10
Graham     21000.00     13
```

Sortér rækker

Det kan være praktisk at få oplysningerne vist i en bestemt rækkefølge. Med ORDER BY-udtrykket kan du sortere oplysningerne efter værdierne i én eller flere kolonner.

I dette eksempel vises medarbejderne i afdeling 84 efter anciennitet:

```
SELECT NAME, JOB, YEARS
   FROM STAFF
   WHERE DEPT = 84
   ORDER BY YEARS
```

Resultatet er:

```
NAME      JOB   YEARS
--------- ----- ------
Davis     Sales      5
Gafney    Clerk      5
Edwards   Sales      7
Quill     Mgr       10
```

Angiv ORDER BY som det sidste udtryk i SELECT-sætningen. De kolonner, der angives i udtrykket, kan være udsagn eller en vilkårlig kolonne i tabellen. Kolonnenavnet i ORDER BY-udtrykket behøver ikke at være blandt dem, du har valgt.

Du kan sortere rækker i stigende eller faldende rækkefølge ved eksplicit at angive ASC eller DESC i ORDER BY-udtrykket. Hvis intet angives, sorteres rækkerne automatisk i stigende rækkefølge. I dette eksempel vises medarbejderne i afdeling 84 i faldende rækkefølge efter anciennitet:

```
SELECT NAME, JOB, YEARS
   FROM STAFF
   WHERE DEPT = 84
   ORDER BY YEARS DESC
```

Resultatet er:

```
NAME      JOB   YEARS
--------- ----- ------
Quill     Mgr       10
Edwards   Sales      7
Davis     Sales      5
Gafney    Clerk      5
```

Du kan både sortere rækker efter tegnværdier og efter numeriske værdier. Her vises medarbejderne i afdeling 84 i alfabetisk rækkefølge efter navn:

```
SELECT NAME, JOB, YEARS
   FROM STAFF
   WHERE DEPT = 84
   ORDER BY NAME
```

Resultatet er:

```
NAME      JOB   YEARS
--------- ----- ------
Davis     Sales      5
Edwards   Sales      7
Gafney    Clerk      5
Quill     Mgr       10
```

Fjern dubletter

Ofte er det overflødigt at få vist de samme oplysninger flere gange. I tabellen STAFF er der f.eks. en DEPT-kolonne, hvor flere afdelingsnumre optræder mere end én gang, og en JOB-kolonner, hvor flere stillingsbetegnelser optræder mere end én gang.

Hvis du ikke vil have vist flere ens rækker, skal du bruge parameteren DISTINCT i SELECT-sætningen. Hvis du indsætter DISTINCT i sætningen, vises hver stilling i en afdeling kun én gang:

```
SELECT DISTINCT DEPT, JOB
   FROM STAFF
   WHERE DEPT < 30
   ORDER BY DEPT, JOB
```

Resultatet er:

```
DEPT   JOB
------ -----
    10 Mgr
    15 Clerk
    15 Mgr
    15 Sales
    20 Clerk
    20 Mgr
    20 Sales
```

Med DISTINCT har du fjernet alle gentagelser fra SELECT-sætningens resultatrækker.

Funktionsrækkefølge

Funktionsrækkefølgen er vigtig. Outputtet frar ét udtryk er input til det næste, som vist nedenfor. I eksemplet “Navngiv udtryk” på side 25 tages der højde for funktionsrækkefølgen.

Nedenstående funktionsrækkefølge er ikke nødvendigvis den samme, som funktionerne udføres på af DB2. Forklaringen er blot en mere intuitiv måde at opfatte forespørgsler på. Funktionerne udføres i denne rækkefølge:

1. FROM-udtryk
2. WHERE-udtryk
3. GROUP BY-udtryk
4. HAVING-udtryk
5. SELECT-udtryk
6. ORDER BY-udtryk

Beregn værdier vha. udtryk

Et *udtryk* er en beregning eller funktion, som medtages i en sætning. I nedenstående sætning beregnes lønnen for hver enkelt medarbejder i afdeling 38, hvis de alle modtog en bonus på 500:

```
SELECT DEPT, NAME, SALARY + 500
   FROM STAFF
   WHERE DEPT = 38
   ORDER BY 3
```

Resultatet er:

```
DEPT   NAME      3
------ --------- ----------------
    38 Abrahams          12509.75
    38 Naughton          13454.75
    38 Quigley           17308.30
    38 Marenghi          18006.75
    38 O'Brien           18506.00
```

Bemærk, at kolonnenavnet for den tredje kolonne er et tal. Nummeret indsættes automatisk, da SALARY+500 ikke er et kolonnenavn. Senere bruges tallet i ORDER BY-udtrykket til at henvise til den tredje kolonne. “Navngiv udtryk” omhandler navngivning af udtryk.

Du kan danne matematiske udtryk vha. de grundlæggende operatorer, dvs. plus (+), minus (–), gange (*) og divideret med (/).

Operatorerne kan bruges på numeriske værdier fra flere typer operander, bl.a.:
- Kolonnenavne (f.eks. RATE * HOURS)
- Konstanter (f.eks. RATE * 1.07)
- Skalarfunktioner (f.eks. LENGTH(NAME) + 1).

Navngiv udtryk

Med det valgfrie AS-udtryk kan du give et udtryk et navn, så det bliver lettere at henvise til udtrykket senere. AS-udtrykket kan bruges til at navngive alle punkter på listen.

I dette eksempel vises alle medarbejdere, hvis løn plus kommission er mindre end 13.000. Udtrykket SALARY + COMM kaldes PAY:

```
SELECT NAME, JOB, SALARY + COMM AS PAY
   FROM STAFF
   WHERE (SALARY + COMM) < 13000
   ORDER BY PAY
```

Resultatet er:

```
NAME      JOB   PAY
--------- ----- ----------
Yamaguchi Clerk   10581.50
Burke     Clerk   11043.50
Scoutten  Clerk   11592.80
Abrahams  Clerk   12246.25
Kermisch  Clerk   12368.60
Ngan      Clerk   12714.80
```

Ved at bruge AS-udtrykket kan du henvise til et bestemt kolonnenavn frem for det nummer, der automatisk blev indsat i ORDER BY-udtrykket. I dette eksempel sammenlignes (SALARY + COMM) - og ikke PAY - med 13000 i WHERE-udtrykket. Det skyldes den rækkefølge, funktionerne udføres i. WHERE-udtrykket vurderes, før (SALARY + COMM) får navnet PAY, fordi SELECT-udtrykket udføres efter WHERE-udtrykket. Derfor kan PAY ikke bruges i prædikatet.

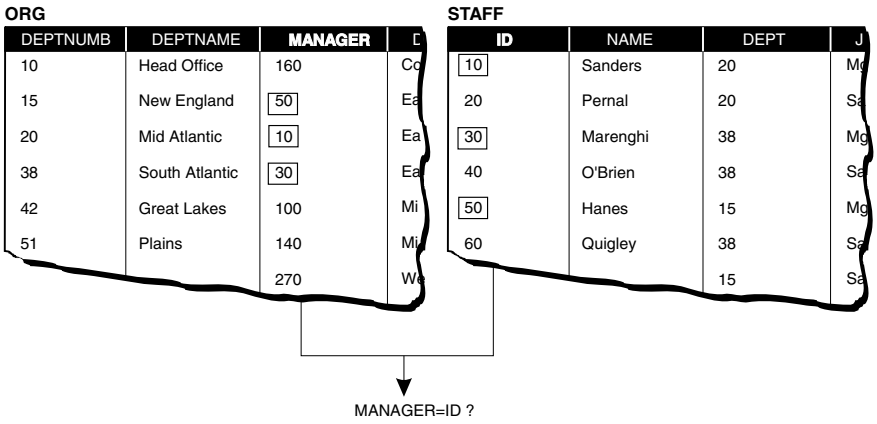
Vælg data fra flere tabeller

Med SELECT-sætningen kan du få vist resultater, der indeholder oplysninger fra to eller flere tabeller. Det kaldes en *sammenkædning*. Du kan f.eks. danne en ny tabel ved at sammenkæde data fra tabellerne STAFF og ORG. Hvis du vil sammenkæde to tabeller, skal du angive de ønskede kolonner i SELECT-udtrykket, tabelnavnene i et FROM-udtryk og søgekriterierne i WHERE-udtrykket. WHERE-udtrykket er valgfrit.

I det næste eksempel knyttes afdelingslederne og afdelingsnavnene sammen. Til det skal du bruge oplysninger fra to tabeller, da medarbejderoplysninger og afdelingsoplysninger opbevares hver for sig (i hhv. STAFF og ORG). Vha. nedenstående forespørgsel kan du vælge kolonnerne NAME og DEPTNAME fra tabellerne STAFF og ORG. Med søgekriterierne begrænses valget til de rækker, hvor værdien i kolonnen MANAGER er lig med værdien i kolonnen ID:

```
SELECT DEPTNAME, NAME
   FROM ORG, STAFF
   WHERE MANAGER = ID
```

I figur 3 på side 27 vises, hvordan kolonner i to forskellige tabeller sammenlignes. De indrammede værdier opfylder søgekriterierne.



Figur 3. Vælg fra tabellerne STAFF og ORG

Resultatet er:

```
DEPTNAME       NAME
-------------- ---------
Mid Atlantic   Sanders
South Atlantic Marenghi
New England    Hanes
Great Lakes    Plotz
Plains         Fraye
Head Office    Molinare
Pacific        Lea
Mountain       Quill
```

I resultatoversigten vises afdelingslederne og deres afdelinger.

Underforespørgsler

Når du skriver en SELECT-sætning, kan du angive flere SELECT-sætninger i WHERE-udtrykket. For hver yderligere SELECT startes en underforespørgsel.

En underforespørgsel kan derefter indeholde endnu en separat underforespørgsel, hvis resultat sættes ind i den oprindelige underforespørgsels WHERE-udtryk. Desuden kan et WHERE-udtryk indeholde underforespørgsler i flere søgekriterier. Underforespørgslen kan henvise til andre tabeller og kolonner end hovedforespørgslen.

Nedenfor vælges division og geografisk placering fra tabellen ORG for den medarbejder, der har id 280 i tabellen STAFF:

```
SELECT DIVISION, LOCATION
   FROM ORG
   WHERE DEPTNUMB = (SELECT DEPT
                        FROM STAFF
                        WHERE ID = 280)
```

Når en sætning behandles, finder DB2 først frem til resultatet af underforespørgslen. Resultatet af underforespørgslen i eksemplet er 66, da medarbejder nr. 280 arbejder i afdeling 66. Det endelige resultat findes så i den række i tabellen ORG, hvor kolonnen DEPTNUMB har værdien 66. Det endelige resultat er:

```
DIVISION   LOCATION
---------- -------------
Western    San Francisco
```

Når du bruger en underforespørgsel, vurderer databasesystemet den og sætter resultatværdien direkte ind i WHERE-udtrykket.

Der er flere oplysninger om underforespørgsler under "Korreleret underforespørgsel" på side 39.

Funktioner

Dette afsnit indeholder en kort beskrivelse af de funktioner, der benyttes i eksemplerne i bogen. En *databasefunktion* er en kobling mellem et sæt inputdataværdier og en resultatværdi.

Funktioner kan være *indbyggede* eller *brugerdefinerede*. DB2 Universal Database har mange indbyggede og forudinstallerede brugerdefinerede funktioner.

De indbyggede funktioner findes i skemaet SYSIBM, og de forudinstallerede brugerdefinerede funktioner findes i skemaet SYSFUN. SYSIBM og SYSFUN er reserverede skemanavne.

De indbyggede og forudinstallerede brugerdefinerede funktioner kan aldrig dække alle brugeres behov. Derfor er det nødvendigt for applikationsudviklere at lave deres egne applikationsspecifikke funktioner. Det kan gøres vha. brugerdefinerede funktioner, så DB2 Universal Database udvides til at omfatte f.eks. specielle virksomhedsspecifikke eller videnskabelige funktioner. Læs mere herom under "Brugerdefinerede funktioner" på side 66.

Beregningsfunktioner

Beregningsfunktioner finder frem til en enkelt resultatværdi ud fra et sæt værdier i en kolonne. Her følger et par eksempler på beregningsfunktioner. Du kan finde en fuldstændig oversigt i *SQL Reference*.

AVG Returnerer gennemsnittet, dvs. summen af værdierne i et sæt divideret med antallet af værdier i sættet

COUNT Returnerer antallet af rækker eller værdier i et sæt rækker eller værdier

MAX Returnerer den højeste værdi i et sæt værdier

MIN Returnerer den laveste værdi i et sæt værdier

I dette eksempel vælges den højeste løn fra tabellen STAFF:

```
SELECT MAX(SALARY)
   FROM STAFF
```

Sætningen returnerer værdien 22959.20 fra tabeleksemplet STAFF.

I det næste eksempel findes navnene på og lønningerne for medarbejdere, der tjener mere end gennemsnittet, selv om de har været ansat i kortere tid end gennemsnitsmedarbejderen.

```
SELECT NAME, SALARY
   FROM STAFF
   WHERE SALARY > (SELECT AVG(SALARY) FROM STAFF)
   AND YEARS < (SELECT AVG(YEARS) FROM STAFF)
```

Resultatet er:

```
 NAME      SALARY
--------- ---------
Marenghi   17506.75
Daniels    19260.25
Gonzales   16858.20
```

I dette eksempel er beregningsfunktionen i WHERE-udtrykket angivet i en *underforespørgsel* i stedet for at blive implementeret direkte (f.eks. WHERE SALARY > AVG(SALARY)). Beregningsfunktioner kan ikke angives i WHERE-udtrykket. Det skyldes den rækkefølge, funktioner udføres i. WHERE-udtrykket vurderes før SELECT-udtrykket. Derfor har beregningsfunktionen ikke adgang til værdisættet, når WHERE-udtrykket vurderes. Værdierne vælges senere vha. SELECT-udtrykket.

Du kan bruge DISTINCT-elementet i parameterværdien til en beregningsfunktion. Dermed elimineres dubletter, før en funktion aktiveres. Når du f.eks. angiver COUNT(DISTINCT WORKDEPT), beregnes antallet af forskellige afdelinger.

Skalarfunktioner

En *skalarfunktion* udfører en funktion med en enkelt værdi, så der returneres en anden enkelt værdi. Her følger et par eksempler på skalarfunktioner i DB2 Universal Database.

ABS Returnerer den absolutte værdi for et tal

HEX Returnerer den hexadecimale repræsentation af en værdi

LENGTH Returnerer antallet af byte i en parameterværdi eller antallet af dobbeltbytetegn i en dobbeltbytestreng

YEAR Udtrækker årstallet fra en værdi for dato/klokkeslæt

Der er en mere detaljeret oversigt over og beskrivelse af skalarfunktioner i *SQL Reference*.

I nedenstående eksempel returneres afdelingsnavnene fra tabellen ORG sammen med længden på de enkelte navne:

```
SELECT DEPTNAME, LENGTH(DEPTNAME)
   FROM ORG
```

Resultatet er:

```
DEPTNAME       2
-------------- -----------
Head Office             11
New England             11
Mid Atlantic            12
South Atlantic          14
Great Lakes             11
Plains                   6
Pacific                  7
Mountain                 8
```

Bemærk: Da AS-udtrykket ikke er brugt til at navngive LENGTH(DEPTNAME), indsættes et nummer automatisk for den anden kolonne.

Tabelfunktioner

Tabelfunktioner returnerer kolonnerne i en tabel, hvilket ligner den tabel, der oprettes vha. en enkel CREATE TABLE-sætning.

En tabelfunktion kan kun bruges i FROM-udtrykket i en SQL-sætning.

Den eneste tabelfunktion, der p.t. understøttes i DB2 Universal Database, er SQLCACHE_SNAPSHOT.

SQLCACHE_SNAPSHOT
Returnerer resultatet af et snapshot af DB2-cachen med dynamiske SQL-sætninger.

Gruppering

I DB2 Universal Database kan data analyseres på grundlag af bestemte kolonner i en tabel.

Du kan organisere rækker vha. en grupperingstruktur, der er defineret i et GROUP BY-udtryk. Den enkleste type *gruppe* er et sæt rækker, der har ens værdier i kolonnen GROUP BY. Kolonnenavnene i SELECT-udtrykket skal enten være grupperingskolonner eller en beregningsfunktion. Beregningsfunktioner returnerer en værdi for hver gruppe i GROUP BY-udtrykket. Hver gruppe repræsenteres af en enkelt række i resultatsættet. I det følgende eksempel vises en oversigt over den højeste løn i hver afdeling:

```
SELECT DEPT, MAX(SALARY) AS MAXIMUM
   FROM STAFF
   GROUP BY DEPT
```

Resultatet er:

```
DEPT   MAXIMUM
------ ---------
    10  22959.20
    15  20659.80
    20  18357.50
    38  18006.00
    42  18352.80
    51  21150.00
    66  21000.00
    84  19818.00
```

Bemærk, at MAX(SALARY) beregnes for hver afdeling, dvs. for en gruppe der er angivet i GROUP BY-udtrykket, ikke for hele selskabet.

Brug af WHERE-udtrykket sammen med GROUP BY-udtrykket

En grupperingsforespørgsel kan indeholde et WHERE-udtryk, som eliminerer irrelevante rækker, før grupperne dannes, og beregningerne foretages. WHERE-udtrykket skal angives *før* GROUP BY-udtrykket. For eksempel:

```
SELECT WORKDEPT, EDLEVEL, MAX(SALARY) AS MAXIMUM
   FROM EMPLOYEE
   WHERE HIREDATE > '1979-01-01'
   GROUP BY WORKDEPT, EDLEVEL
   ORDER BY WORKDEPT, EDLEVEL
```

Resultatet er:

```
WORKDEPT EDLEVEL MAXIMUM
-------- ------- -----------
D11           17    18270.00
D21           15    27380.00
D21           16    36170.00
D21           17    28760.00
E11           12    15340.00
E21           14    26150.00
```

Bemærk, at de kolonnenavne, der angives i SELECT-sætningen, også er med i GROUP BY-udtrykket. Hvis et kolonnenavn ikke angives begge steder, opstår der en fejl. GROUP BY-udtrykket returnerer en række for hver entydig kombination af WORKDEPT og EDLEVEL.

Brug af HAVING-udtrykket efter GROUP BY-udtrykket

Du kan angive en kvalificerende betingelse for grupper, så DB2 kun returnerer et resultat for de grupper, der opfylder betingelsen. Det gør du ved at inkludere et HAVING-udtryk *efter* GROUP BY-udtrykket. Et HAVING-udtryk kan

indeholde ét eller flere prædikater, der er forbundet med AND eller OR. Hvert prædikat sammenligner en egenskab for gruppen, f.eks. AVG(SALARY) med én af følgende:

- En anden egenskab for gruppen

  Eksempel:

  ```
  HAVING AVG(SALARY) > 2 * MIN(SALARY)
  ```

- En konstant

  For eksempel:

  ```
  HAVING AVG(SALARY) > 20000
  ```

I dette eksempel vises den højeste og laveste løn i afdelinger med over 4 medarbejdere:

```
SELECT WORKDEPT, MAX(SALARY) AS MAXIMUM, MIN(SALARY) AS MINIMUM
   FROM EMPLOYEE
   GROUP BY WORKDEPT
   HAVING COUNT(*) > 4
   ORDER BY WORKDEPT
```

Resultatet er:

```
WORKDEPT MAXIMUM     MINIMUM
-------- ----------- -----------
D11         32250.00    18270.00
D21         36170.00    17250.00
E11         29750.00    15340.00
```

Det er muligt, men ikke almindeligt, for en forespørgsel at indeholde et HAVING-udtryk uden et GROUP BY-udtryk. I det tilfælde betragter DB2 hele tabellen som én gruppe. Fordi tabellen betragtes som én gruppe, kan du højst få vist én resultatrække. Hvis HAVING-betingelsen er sand for tabellen som helhed, vises det valgte resultat, som udelukkende består af beregningsfunktioner, ellers returneres ingen rækker.

Kapitel 5. Udtryk og underforespørgsler

DB2 er fleksibel mht. udformning af forespørgsler. I dette kapitel beskrives nogle vigtige metoder til udformning af komplekse forespørgsler.

Kapitlet behandler følgende emner:

- Skalare fullselect
- Konvertér datatyper
- CASE-udtryk
- Tabeludtryk
- Korrelationsnavn

Skalare fullselect

En fullselect er en form for forespørgsel, der kan bruges i SQL-sætninger. En skalar fullselect er en fullselect, som returnerer én række med én kolonneværdi. Skalare fullselect bruges til at hente dataværdier fra databasen, så de kan bruges i et udtryk.

- I dette eksempel vises navnene på de medarbejdere, hvis løn ligger over gennemsnitslønnen for alle medarbejdere. Den skalare fullselect i forespørgslen er SELECT-sætningen i parentes.

```
SELECT LASTNAME, FIRSTNME
   FROM EMPLOYEE
   WHERE  SALARY > (SELECT AVG(SALARY)
                      FROM EMPLOYEE)
```

- I dette eksempel findes gennemsnitslønnen for medarbejderne i STAFF-tabellen og gennemsnitslønnen for medarbejderne i EMPLOYEE-tabellen.

```
SELECT AVG(SALARY) AS "Average_Employee",
   (SELECT AVG(SALARY) AS "Average_Staff" FROM STAFF)
      FROM EMPLOYEE
```

Konvertér datatyper

Det kan nogle gange være nødvendigt at konvertere værdier fra én datatype til en anden, f.eks. fra en numerisk værdi til en tegnstreng. Hvis du vil konvertere en værdi til en anden type, skal du bruge CAST-specifikationen.

CAST kan også bruges til at afkorte meget lange tegnstrenge. I tabellen EMP_RESUME er kolonnen RESUME af typen CLOB(5K). Lad os antage, at du kun vil have vist de først 370 tegn med personlige oplysninger om den pågæl-

dende. Hvis du kun vil have vist de først 370 tegn i ASCII-format i CV'erne i tabellen EMP_RESUME, skal du skrive:

```
SELECT EMPNO, CAST(RESUME AS VARCHAR(370))
   FROM EMP_RESUME
   WHERE RESUME_FORMAT = 'ascii'
```

Du får vist en advarsel om, at værdier over 370 tegn afkortes.

Du kan konvertere NULL-værdier til andre datatyper, som er mere hensigtsmæssige i en forespørgsel. Under "Fælles tabeludtryk" på side 36 vises et eksempel på konvertering med dette formål.

CASE-udtryk

Du kan bruge CASE-udtryk i SQL-sætninger til at behandle datarepræsentationen i en tabel. Dermed kan du benytte betingede udsagn, der minder om CASE-sætninger i visse programmeringssprog.

- Hvis du vil ændre afdelingsnumrene i kolonnen DEPTNAME i tabellen ORG til meningsfyldte ord, skal du skrive:

  ```
  SELECT DEPTNAME,
     CASE DEPTNUMB
       WHEN 10 THEN 'Marketing'
       WHEN 15 THEN 'Research'
       WHEN 20 THEN 'Development'
       WHEN 38 THEN 'Accounting'
       ELSE 'Sales'
     END  AS FUNCTION
     FROM ORG
  ```

  Resultatet er:

  ```
  DEPTNAME       FUNCTION
  -------------- -----------
  Head Office    Marketing
  New England    Research
  Mid Atlantic   Development
  South Atlantic Accounting
  Great Lakes    Sales
  Plains         Sales
  Pacific        Sales
  Mountain       Sales
  ```

- Du kan bruge CASE-udtryk som beskyttelse mod fejl som f.eks. division med nul. I dette eksempel undgås en fejl, fordi sætningsbetingelsen forhindrer, at divisionen udføres, hvis medarbejderen hverken får bonus eller provision:

  ```
  SELECT LASTNAME, WORKDEPT FROM EMPLOYEE
     WHERE(CASE
              WHEN BONUS+COMM=0 THEN NULL
              ELSE SALARY/(BONUS+COMM)
           END ) > 10
  ```

- Du kan bruge et CASE-udtryk til at beregne forholdet mellem summen af en række værdier fra en kolonne og summen af alle værdier i kolonnen. Forholdet kan indeholdes i en enkelt sætning, der indeholder et CASE-udtryk. Udtrykket bevirker, at det kun nødvendigt at gennemløbe data én gang. Uden et CASE-udtryk er mindst to gennemløb nødvendige ved udførelse af den samme beregning.

  I dette eksempel beregnes forholdet mellem summen af lønningerne i afdeling 20 og den samlede lønudbetaling vha. et CASE-udtryk:

```
SELECT CAST(CAST (SUM(CASE
                        WHEN DEPT = 20 THEN SALARY
                        ELSE 0
                      END) AS DECIMAL(7,2))/
                    SUM(SALARY) AS DECIMAL (3,2))
  FROM STAFF
```

  Resultatet er 0.11. Bemærk, at resultatets præcision bevares, når CAST-funktionerne angives.

- Du kan bruge et CASE-udtryk til at vurdere en enkel funktion i stedet for at kalde selve funktionen, hvilket ville medføre et tidstillæg. Eksempel:

```
CASE
   WHEN X<0 THEN -1
   WHEN X=0 THEN 0
   WHEN X>0 THEN 1
END
```

  Dette udtryk giver samme resultat som den brugerdefinerede funktion SIGN i skemaet SYSFUN.

Tabeludtryk

Hvis du kun skal bruge definitionen af et udpluk til en enkelt forespørgsel, kan du bruge et *tabeludtryk*.

Tabeludtryk er midlertidige og eksisterer kun, indtil SQL-sætningen er udført. De kan ikke deles som udpluk, men de er mere fleksible end udpluk.

I dette afsnit beskrives brugen af fælles tabeludtryk og indflettede tabeludtryk i forespørgsler.

Indflettet tabeludtryk

Et indflettet tabeludtryk er et midlertidigt udpluk, hvor definitionen er *indflettet* (defineret direkte) i FROM-udtrykket i hovedforespørgslen.

I nedenstående eksempel bruges et indflettet tabeludtryk til at finde den gennemsnitlige samlede aflønning, uddannelsesniveau og ansættelsesår for medarbejdere, hvis uddannelsesniveau er højere end 16:

```
SELECT EDLEVEL, HIREYEAR, DECIMAL(AVG(TOTAL_PAY),7,2)
   FROM (SELECT EDLEVEL, YEAR(HIREDATE) AS HIREYEAR,
                SALARY+BONUS+COMM AS TOTAL_PAY
           FROM EMPLOYEE
           WHERE EDLEVEL > 16)  AS PAY_LEVEL
   GROUP BY EDLEVEL, HIREYEAR
   ORDER BY EDLEVEL, HIREYEAR
```

Resultatet er:

```
EDLEVEL HIREYEAR    3
------- ----------- ---------
     17        1967  28850.00
     17        1973  23547.00
     17        1977  24430.00
     17        1979  25896.50
     18        1965  57970.00
     18        1968  32827.00
     18        1973  45350.00
     18        1976  31294.00
     19        1958  51120.00
     20        1975  42110.00
```

Forespørgslen bruger et indflettet tabeludtryk til først at finde ansættelsesåret i kolonnen HIREDATE, så det efterfølgende kan bruges i GROUP BY-udtrykket. Du vil måske ikke oprette et udpluk, hvis du senere skal lave lignende forespørgsler med andre værdier for EDLEVEL.

Den skalare indbyggede funktion DECIMAL bruges i dette eksempel. DECIMAL returnerer en decimalrepræsentation af et tal eller en tegnstreng. Der er flere oplysninger om funktioner i *SQL Reference*.

Fælles tabeludtryk

Et *fælles tabeludtryk* er et tabeludtryk, du opretter, så det kan bruges hele vejen igennem en kompleks forespørgsel. Definér og navngiv den i begyndelsen af forespørgslen vha. et WITH-udtryk. Ved gentagne henvisninger til et fælles tabeludtryk bruges de samme resultatrækker. Hvis du derimod bruger indflettede tabeludtryk eller udpluk, genereres resultatrækkerne hver gang, hvilket kan give forskellige resultater.

I det følgende eksempel vises alle de personer i selskabet, hvis uddannelsesniveau er højere end 16, som tjener mindre i gennemsnit end dem, der blev ansat samme år, og som har samme uddannelse. De enkelte dele af forespørgslen beskrives nærmere nedenfor.

1

```
WITH
      PAYLEVEL AS
         (SELECT EMPNO, EDLEVEL, YEAR(HIREDATE) AS HIREYEAR,
                 SALARY+BONUS+COMM AS TOTAL_PAY
             FROM EMPLOYEE
             WHERE EDLEVEL > 16),
```

2

```
      PAYBYED (EDUC_LEVEL, YEAR_OF_HIRE, AVG_TOTAL_PAY) AS
         (SELECT EDLEVEL, HIREYEAR, AVG(TOTAL_PAY)
             FROM PAYLEVEL
             GROUP BY EDLEVEL, HIREYEAR)
```

3

```
SELECT EMPNO, EDLEVEL, YEAR_OF_HIRE, TOTAL_PAY, DECIMAL(AVG_TOTAL_PAY,7,2)
   FROM PAYLEVEL, PAYBYED
   WHERE EDLEVEL = EDUC_LEVEL
     AND HIREYEAR = YEAR_OF_HIRE
     AND TOTAL_PAY < AVG_TOTAL_PAY
```

1 Dette er et fælles tabeludtryk med navnet PAYLEVEL. Resultattabellen indeholder medarbejderens personalenummer, ansættelsesår, samlede aflønning og uddannelsesniveau. Kun de rækker, hvor medarbejderens uddannelsesniveau er højere end 16, vises.

2 Dette er et fælles tabeludtryk med navnet PAYBYED (PAY BY EDucation). Det bruger tabellen PAYLEVEL, som blev oprettet i det foregående fælles tabeludtryk, til at finde uddannelsesniveau, ansættelsesår og gennemsnitlig aflønning for de medarbejdere inden for hvert uddannelsesniveau, som er ansat samme år. De kolonner, der returneres i tabellen har fået andre navne end kolonnenavnene i listen over valgte kolonner, f.eks. EDUC_LEVEL. Dermed fås resultatrækkerne PAYBYED, som er identisk med resultatet af eksemplet under Indflettet tabeludtryk.

3 Endelig kommer den faktiske forespørgsel, som giver det ønskede resultat. De to tabeller (PAYLEVEL, PAYBYED) sammenkædes, så det er muligt at finde frem til de personer, som har en samlet aflønning, der ligger under gennemsnittet for dem, der er ansat samme år. Bemærk, at PAYBYED er baseret på PAYLEVEL. Så PAYLEVEL læses faktisk to gange i den fuldstændige sætning. Begge gange bruges samme rækker til at vurdere forespørgslen.

Det endelige resultat er:

```
EMPNO  EDLEVEL YEAR_OF_HIRE TOTAL_PAY     5
------ ------- ------------ ------------- ---------
000210      17         1979      20132.00  25896.50
```

Korrelationsnavn

Et *korrelationsnavn* er en kvalifikator, der bruges til at adskille forskellige måder at bruge et objekt. Et korrelationsnavn kan defineres i FROM-udtrykket i en forespørgsel og i det første udtryk i en UPDATE- eller DELETE-sætning. Det kan knyttes til en tabel, et udpluk eller et indflettet tabeludtryk, men kun inden for den sammenhæng, hvor det er defineret.

Således fastsætter udtrykket FROM STAFF S, ORG O bogstaverne S og O som korrelationsnavne til hhv. STAFF og ORG.

```
SELECT NAME, DEPTNAME
   FROM STAFF S, ORG O
   WHERE O.MANAGER = S.ID
```

Når du har defineret et korrelationsnavn, kan du *kun* bruge korrelationsnavnet til at kvalificere objektet. Hvis du i ovenstående eksempel havde skrevet ORG.MANAGER=STAFF.ID, kan sætningen ikke udføres.

Korrelationsnavnet kan også bruges som et kort navn på et databaseobjekt. Det er lettere at skrive S end at skrive STAFF.

Med korrelationsnavne kan du lave **dubletter** af et objekt. Det er nyttigt, hvis du skal lave sammenligninger inden for tabellen. I nedenstående eksempel sammenlignes tabellen EMPLOYEE med en anden forekomst af samme tabel for at finde frem til alle medarbejderes afdelingsledere. Du får vist navnene på de medarbejdere, hvis stilling ikke er 'designer', samt deres afdelingsleder og afdelingsnummer.

```
SELECT E2.FIRSTNME, E2.LASTNAME, E2.JOB, E1.FIRSTNME AS MGR_FIRSTNAME,
                    E1.LASTNAME AS MGR_LASTNAME, E1.WORKDEPT
   FROM EMPLOYEE E1, EMPLOYEE E2
   WHERE E1.WORKDEPT = E2.WORKDEPT
     AND E1.JOB = 'MANAGER'
     AND E2.JOB <> 'MANAGER'
     AND E2.JOB <> 'DESIGNER'
```

Resultatet er:

```
FIRSTNME     LASTNAME   JOB      MGR_FIRSTNAME MGR_LASTNAME    WORKDEPT
------------ ---------- -------- ------------- --------------- --------
DOLORES      QUINTANA   ANALYST  SALLY         KWAN            C01
HEATHER      NICHOLLS   ANALYST  SALLY         KWAN            C01
JAMES        JEFFERSON  CLERK    EVA           PULASKI         D21
SALVATORE    MARINO     CLERK    EVA           PULASKI         D21
```

```
DANIEL      SMITH      CLERK    EVA          PULASKI        D21
SYBIL       JOHNSON    CLERK    EVA          PULASKI        D21
MARIA       PEREZ      CLERK    EVA          PULASKI        D21
ETHEL       SCHNEIDER  OPERATOR EILEEN       HENDERSON      E11
JOHN        PARKER     OPERATOR EILEEN       HENDERSON      E11
PHILIP      SMITH      OPERATOR EILEEN       HENDERSON      E11
MAUDE       SETRIGHT   OPERATOR EILEEN       HENDERSON      E11
RAMLAL      MEHTA      FIELDREP THEODORE     SPENSER        E21
WING        LEE        FIELDREP THEODORE     SPENSER        E21
JASON       GOUNOT     FIELDREP THEODORE     SPENSER        E21
```

Korreleret underforespørgsel

En underforespørgsel, der kan henvise til en af de ovenfor nævnte tabeller, kaldes en *korreleret underforespørgsel*. Man kan også sige, at underforespørgslen har en *korreleret reference* til en tabel i hovedforespørgslen.

Nedenstående eksempel bruger en ukorreleret underforespørgsel til at få vist personalenummer og navn på medarbejdere i afdeling 'A00', hvis løn er højere end afdelingsgennemsnittet:

```
SELECT EMPNO, LASTNAME
   FROM EMPLOYEE
   WHERE WORKDEPT = 'A00'
     AND SALARY > (SELECT AVG(SALARY)
                     FROM EMPLOYEE
                     WHERE WORKDEPT = 'A00')
```

Resultatet er:

```
EMPNO  LASTNAME
------ ---------------
000010 HAAS
000110 LUCCHESSI
```

Hvis du vil have vist gennemsnitslønnen for hver afdeling, skal underforespørgslen vurderes én gang for hver afdeling. Det kan du gøre vha. korrelationsegenskaben i SQL, som gør det muligt at skrive en underforespørgsel, som udføres gentagne gange, én for hver række i den tabel, der er angivet i den yderste forespørgsel.

I følgende eksempel bruges en korreleret underforespørgsel til at få vist alle medarbejdere, hvis løn er højere end gennemsnittet i afdelingen:

```
SELECT E1.EMPNO, E1.LASTNAME, E1.WORKDEPT
   FROM EMPLOYEE E1
   WHERE SALARY > (SELECT AVG(SALARY)
                     FROM EMPLOYEE E2
                     WHERE E2.WORKDEPT = E1.WORKDEPT)
   ORDER BY E1.WORKDEPT
```

I denne forespørgsel vurderes underforespørgslen én gang for hver afdeling. Resultatet er:

```
EMPNO  LASTNAME        WORKDEPT
------ --------------- --------
000010 HAAS            A00
000110 LUCCHESSI       A00
000030 KWAN            C01
000060 STERN           D11
000150 ADAMSON         D11
000170 YOSHIMURA       D11
000200 BROWN           D11
000220 LUTZ            D11
000070 PULASKI         D21
000240 MARINO          D21
000270 PEREZ           D21
000090 HENDERSON       E11
000280 SCHNEIDER       E11
000100 SPENSER         E21
000330 LEE             E21
000340 GOUNOT          E21
```

Hvis du vil skrive en forespørgsel med en korreleret underforespørgsel, skal du bruge samme grundlæggende format som til en ydre forespørgsel med en underforespørgsel. I FROM-udtrykket i den ydre forespørgsel - lige efter tabelnavnet - skal du imidlertid anbringe korrelationsnavnet. Underforespørgslen kan så indeholde kolonnehenvisninger, der er kvalificeret vha. korrelationsnavnet. Hvis f.eks. E1 er et korrelationsnavn, så betyder E1.WORKDEPT værdien WORKDEPT i den aktuelle række i tabellen i den ydre forespørgsel. Underforespørgslen revurderes (principielt) for hver række i tabellen i den ydre forespørgsel.

Når du bruger en korreleret underforespørgsel, gør systemet arbejdet for dig, og du reducerer den mængde kode, der skal skrives i applikationen.

DB2 tillader også ukvalificerede korrelerede referencer. Eksempelvis har tabellen EMPLOYEE en kolonne, der hedder LASTNAME, og tabellen SALES har en kolonne, der hedder SALES_PERSON, men ingen kolonne, der hedder LASTNAME.

```
SELECT LASTNAME, FIRSTNME, COMM
   FROM EMPLOYEE
   WHERE 3 > (SELECT AVG(SALES)
                FROM SALES
                WHERE LASTNAME = SALES_PERSON)
```

I dette tilfælde søger systemet efter kolonnen LASTNAME i det inderste FROM-udtryk. Den findes ikke, hvorefter det næstinderste FROM-udtryk søges. I dette tilfælde er det lig med det ydre FROM-udtryk. Selv om det ikke altid er nødvendigt, anbefales det at kvalificere korrelerede referencer for at forbedre forespørgslens læsbarhed og sikre, at det ønskede resultat opnås.

Udfør korreleret underforespørgsel

Hvornår skal du bruge en korreleret underforespørgsel? Brugen af en beregningsfunktion kan være en indikator.

Forestil dig, at du vil have vist de medarbejdere, hvis uddannelsesniveau ligger højere end afdelingsgennemsnittet.

Først skal du finde frem til de kolonner, der skal vælges. Opgaven lyder "Vis medarbejdere". Det indebærer, at kolonnen LASTNAME i tabellen EMPLOYEE skulle være nok til at identificere medarbejdere entydigt. Opgaven indeholder også betingelserne uddannelsesniveau (EDLEVEL) og medarbejdernes afdelinger (WORKDEPT). Du skal ikke eksplicit have vist disse kolonner, men resultatet kan blive lettere at få overblik over, hvis du vælger dem. Du kan nu udforme en del af forespørgslen:

```
SELECT LASTNAME, WORKDEPT, EDLEVEL
   FROM EMPLOYEE
```

Dernæst skal du bruge et søgekriterium, dvs. et WHERE-udtryk. Opgaven lyder "...hvis uddannelsesniveau ligger højere end afdelingsgennemsnittet". Det betyder, at for hver medarbejder i tabellen skal det gennemsnitlige uddannelsesniveau i afdelingen udregnes. Denne sætning svarer til beskrivelsen af en korreleret underforespørgsel. Der beregnes en ukendt egenskab for hver række, nemlig det gennemsnitlige uddannelsesniveau for den aktuelle medarbejders afdeling. Du skal bruge et korrelationsnavn til tabellen EMPLOYEE:

```
SELECT LASTNAME, WORKDEPT, EDLEVEL
   FROM EMPLOYEE E1
```

Underforespørgslen er enkel. Den beregner det gennemsnitlige uddannelsesniveau for hver afdeling. Den fuldstændige SQL-sætning ser således ud:

```
SELECT LASTNAME, WORKDEPT, EDLEVEL
   FROM EMPLOYEE E1
   WHERE EDLEVEL > (SELECT AVG(EDLEVEL)
                       FROM EMPLOYEE  E2
                       WHERE E2.WORKDEPT = E1.WORKDEPT)
```

Resultatet er:

```
LASTNAME        WORKDEPT EDLEVEL
--------------- -------- -------
HAAS            A00           18
KWAN            C01           20
PULASKI         D21           16
HENDERSON       E11           16
LUCCHESSI       A00           19
PIANKA          D11           17
SCOUTTEN        D11           17
JONES           D11           17
LUTZ            D11           18
MARINO          D21           17
```

```
JOHNSON         D21           16
SCHNEIDER       E11           17
MEHTA           E21           16
GOUNOT          E21           16
```

Måske vil du hellere have vist medarbejderens afdelingsnavn end -nummer. De oplysninger, du skal bruge (DEPTNAME) ligger i en særskilt tabel (DEPARTMENT). Den ydre forespørgsel, som definerer en korrelationsvariabel, kan også være en sammenkædningsforespørgsel. Læs nærmere herom under "Vælg data fra flere tabeller" på side 26.

Når du bruger sammenkædning i en ydre forespørgsel, skal du angive de tabeller, der skal sammenkædes, i FROM-udtrykket og anbringe korrelationsnavnet ved siden af det relevante tabelnavn.

Hvis du vil ændre forespørgslen, så du får vist afdelingsnavnet i stedet for -nummeret, skal du udskifte WORKDEPT med DEPTNAME i listen over valgte kolonner. FROM-udtrykket skal nu også indeholde tabellen DEPARTMENT, og WHERE-udtrykket skal angive den relevante sammenkædningsbetingelse.

Sådan ser den ændrede forespørgsel ud:

```
SELECT LASTNAME, DEPTNAME, EDLEVEL
   FROM EMPLOYEE E1, DEPARTMENT
   WHERE E1.WORKDEPT = DEPARTMENT.DEPTNO
   AND EDLEVEL > (SELECT AVG(EDLEVEL)
                  FROM EMPLOYEE E2
                  WHERE E2.WORKDEPT = E1.WORKDEPT)
```

Resultatet er:

```
LASTNAME        DEPTNAME                    EDLEVEL
--------------- --------------------------- -------
HAAS            SPIFFY COMPUTER SERVICE DIV.      18
LUCCHESSI       SPIFFY COMPUTER SERVICE DIV.      19
KWAN            INFORMATION CENTER                20
PIANKA          MANUFACTURING SYSTEMS             17
SCOUTTEN        MANUFACTURING SYSTEMS             17
JONES           MANUFACTURING SYSTEMS             17
LUTZ            MANUFACTURING SYSTEMS             18
PULASKI         ADMINISTRATION SYSTEMS            16
MARINO          ADMINISTRATION SYSTEMS            17
JOHNSON         ADMINISTRATION SYSTEMS            16
HENDERSON       OPERATIONS                        16
SCHNEIDER       OPERATIONS                        17
MEHTA           SOFTWARE SUPPORT                  16
GOUNOT          SOFTWARE SUPPORT                  16
```

Ovenstående eksempler viser, at korrelationsnavnet i en underforespørgsel skal defineres i FROM-udtrykket i en forespørgsel, som indeholder den korrelerede underforespørgsel. Det kan medføre indfletning i flere lag.

Nogle afdelinger har kun få medarbejdere, og derfor kan det gennemsnitlige uddannelsesniveau være misvisende. Du kan derfor vedtage, at hvis det skal give mening at sammenligne en medarbejders uddannelsesniveau med afdelingsgennemsnittet, skal der være mindst fem ansatte i afdelingen. Opgaven lyder nu: vis de medarbejdere, hvis uddannelsesniveau ligger højere end afdelingsgennemsnittet, men se bort fra afdelinger med færre end fem medarbejdere.

Det kræver endnu en underforespørgsel, da der for hver medarbejder i den ydre forespørgsel skal beregnes, hvor mange ansatte der er i vedkommendes afdeling:

```
SELECT COUNT(*)
   FROM EMPLOYEE E3
   WHERE E3.WORKDEPT = E1.WORKDEPT
```

Kun hvis antallet er større end eller lig med 5, beregnes et gennemsnit:

```
SELECT AVG(EDLEVEL)
   FROM EMPLOYEE E2
   WHERE E2.WORKDEPT = E1.WORKDEPT
   AND 5 <= (SELECT COUNT(*)
                FROM EMPLOYEE  E3
                WHERE E3.WORKDEPT = E1.WORKDEPT)
```

Endelig skal vi kun have vist de medarbejdere, hvis uddannelse ligger over afdelingsgennemsnittet:

```
SELECT LASTNAME, DEPTNAME, EDLEVEL
   FROM EMPLOYEE E1, DEPARTMENT
   WHERE E1.WORKDEPT = DEPARTMENT.DEPTNO
   AND EDLEVEL >
   (SELECT AVG(EDLEVEL)
       FROM EMPLOYEE E2
       WHERE E2.WORKDEPT = E1.WORKDEPT
       AND 5 <=
       (SELECT COUNT(*)
           FROM EMPLOYEE E3
           WHERE E3.WORKDEPT = E1.WORKDEPT))
```

Resultatet er:

```
LASTNAME        DEPTNAME                    EDLEVEL
--------------- --------------------------- -------
PIANKA          MANUFACTURING SYSTEMS            17
SCOUTTEN        MANUFACTURING SYSTEMS            17
JONES           MANUFACTURING SYSTEMS            17
LUTZ            MANUFACTURING SYSTEMS            18
PULASKI         ADMINISTRATION SYSTEMS           16
MARINO          ADMINISTRATION SYSTEMS           17
JOHNSON         ADMINISTRATION SYSTEMS           16
HENDERSON       OPERATIONS                       16
SCHNEIDER       OPERATIONS                       17
```

Kapitel 6. Brug af operatorer og prædikater i forespørgsler

I DB2 Universal Database kan du kombinere forespørgsler med forskellige *sammenkædningsoperator* og konstruere komplekse betingede sætninger med *kvantificerede prædikater*.

I dette kapitel lærer du, hvordan du kan:

- Kombinere forskellige tabeller med sammenkædningsoperatorerne UNION, EXCEPT og INTERSECT.
- Konstruere komplekse betingelser til forespørgsler med kvantificerede prædikater. Grundlæggende prædikater blev kort gennemgået under "Vælg række" på side 20.

Kombinér forespørgsler vha. sammenkædningsoperatorer

Med sammenkædningsoperatorerne UNION, EXCEPT og INTERSECT kan du kombinere to eller flere ydre forespørgsler til én forespørgsel. Hver forespørgsel, som sammenkædningsoperatorne forbinder, udføres, og resultaterne kombineres. Hver operator giver et forskelligt resultat.

Operatoren UNION

UNION-operatoren udleder en resultattabel ved at kombinere to andre resultattabeller, f.eks. TABEL1 og TABEL2, og eliminere eventuelle dubletter i tabellerne. Hvis du bruger ALL sammen med UNION, dvs. UNION ALL, elimineres dubletterne ikke. I begge tilfælde stammer hver række i den afledte tabel fra TABEL1 eller TABEL2.

I nedenstående eksempel på UNION-operatoren vises navnet på alle medarbejdere, hvis løn er højere end 21.000, eller som er ledere og har været ansat i under 8 år:

```
1
SELECT ID, NAME FROM STAFF WHERE SALARY > 21000
UNION

2
SELECT ID, NAME FROM STAFF WHERE JOB='Mgr' AND YEARS < 8
ORDER BY ID
```

Resultatet af de enkelte forespørgsler er:

1

```
ID     NAME
------ ---------
   140 Fraye
   160 Molinare
   260 Jones
```

2

```
ID     NAME
------ ---------
    10 Sanders
    30 Marenghi
   100 Plotz
   140 Fraye
   160 Molinare
   240 Daniels
```

Databasesystemet kombinerer resultaterne af de to forespørgsler, eliminerer dubletterne og viser det endelige resultat i stigende rækkefølge.

```
ID     NAME
------ ---------
    10 Sanders
    30 Marenghi
   100 Plotz
   140 Fraye
   160 Molinare
   240 Daniels
   260 Jones
```

Hvis du bruger ORDER BY-udtrykket i en forespørgsel sammen med en sammenkædningsoperator, skal du skrive den efter den sidste forespørgsel. Systemet sorterer de kombinerede resultatrækker.

Hvis kolonnenavnene i de to tabeller ikke er ens, vises der i den kombinerede resultattabel ikke noget navn på de tilsvarende kolonner. I stedet nummereres kolonnerne i den rækkefølge, de vises i. Med andre ord, hvis resultattabellen skal sorteres, skal du angive kolonnenummeret i ORDER BY-udtrykket.

Operatoren EXCEPT

EXCEPT-operatoren udleder en resultattabel ved at medtage alle de rækker, som findes i TABEL1, men ikke i TABEL2, og eliminere alle dubletter. Hvis du bruger ALL sammen med EXCEPT, dvs. EXCEPT ALL, elimineres dubletterne ikke.

I nedenstående eksempel på EXCEPT-operatoren vises navnet på alle medarbejdere, som tjener mere end 21.000, men som ikke har en lederstilling, og som har været ansat i 8 år eller længere.

```
SELECT ID, NAME FROM STAFF WHERE SALARY > 21000
EXCEPT
SELECT ID, NAME FROM STAFF WHERE JOB='Mgr' AND YEARS < 8
```

Resultatet af de enkelte forespørgsler vises i afsnittet om UNION. Ovenstående sætning giver følgende resultat:

```
ID     NAME
------ ---------
   260 Jones
```

Operatoren INTERSECT

INTERSECT-operatoren udleder en resultattabel ved kun at medtage de rækker, som findes i både TABLE1 og TABLE2 og eliminere alle dubletter. Hvis du bruger ALL sammen med INTERSECT, dvs. INTERSECT ALL, elimineres dubletterne ikke.

I nedenstående eksempel på INTERSECT-operatoren vises navn og id på medarbejdere, som tjener mere end 21.000, har en lederstilling og har været ansat i under 8 år.

```
SELECT ID, NAME FROM STAFF WHERE SALARY > 21000
INTERSECT
SELECT ID, NAME FROM STAFF WHERE JOB='Mgr' AND YEARS < 8
```

Resultatet af de enkelte forespørgsler vises i afsnittet om UNION. Resultatet af de to forespørgsler med INTERSECT er:

```
ID     NAME
------ ---------
   140 Fraye
   160 Molinare
```

Når du bruger operatorerne UNION, EXCEPT og INTERSECT, skal du huske følgende:

- Alle elementer, der modsvarer hinanden i de valgte kolonner i forespørgslerne til operatorerne, skal være kompatible. Der er flere oplysninger i oversigten over datatypekompatibilitet i *SQL Reference*.
- Hvis du bruger et ORDER BY-udtryk, skal det placeres efter den sidste forespørgsel med en sammenkædningsoperator. Kolonnenavnet kan kun bruges i ORDER BY-udtrykket, hvis det er identisk med de tilsvarende elementer i SELECT-listen i forespørgslerne for hver operator.
- Funktioner mellem kolonner med samme datatype og samme længde resulterer i en kolonne med den type og længde. Du kan læse om resultatdatatyper i *SQL Reference*, hvis du vil vide mere om resultaterne af sammenkædningsoperatorerne UNION, EXCEPT og INTERSECT.

Prædikater

Med prædikater kan du angive betingelser, som skal opfyldes, for at en række behandles. Grundlæggende prædikater beskrives under "Vælg række" på side 20. IN, BETWEEN, LIKE, EXISTS og kvantificerede prædikater beskrives i dette afsnit.

Prædikatet IN

Brug prædikatet IN til at sammenligne en værdi med flere andre værdier. Eksempel:

```
SELECT NAME
   FROM STAFF
   WHERE DEPT IN (20, 15)
```

Dette eksempel svarer til:

```
SELECT NAME
   FROM STAFF
   WHERE DEPT = 20  OR DEPT = 15
```

Du kan bruge prædikaterne IN og NOT IN, når en underforespørgsel returnerer et sæt værdier. I dette eksempel vises efternavnene på de medarbejdere, der er ansvarlige for projekt MA2100 og OP2012:

```
SELECT LASTNAME
   FROM EMPLOYEE
   WHERE EMPNO IN
      (SELECT RESPEMP
          FROM PROJECT
          WHERE PROJNO = 'MA2100'
          OR PROJNO = 'OP2012')
```

Underforespørgslen vurderes én gang, og resultatet sættes inde i den ydre forespørgsel. Hvis ovenstående underforespørgsel f.eks. vælger personalenumrene 10 og 330, vurderes den ydre forespørgsel, som om dens WHERE-udtryk så sådan ud:

```
WHERE EMPNO IN (10, 330)
```

Underforespørgslen kan returnere ingen, en eller flere værdier.

Prædikatet BETWEEN

BETWEEN-prædikatet sammenligner en enkelt værdi med et interval af værdier.

I dette eksempel vises navnet på de medarbejdere, som tjener mellem 10.000 og 20.000:

```
SELECT LASTNAME
   FROM EMPLOYEE
   WHERE SALARY BETWEEN 10000 AND 20000
```

Det svarer til:

```
SELECT LASTNAME
   FROM EMPLOYEE
   WHERE SALARY >= 10000 AND SALARY <= 20000
```

I det næste eksempel vises de medarbejdere, som tjener under 10.000 eller over 20.000:

```
SELECT LASTNAME
   FROM EMPLOYEE
   WHERE SALARY NOT BETWEEN 10000 AND 20000
```

Prædikatet LIKE

Brug prædikatet LIKE til at søge efter strenge med bestemte mønstre. Mønstret angives vha. procenttegn og understregningstegn.

- Understregningstegnet (_) repræsenterer et vilkårligt enkelt tegn.
- Procenttegnet (%) repræsenterer en streng med nul eller flere tegn.
- Alle andre tegn repræsenterer sig selv.

I dette eksempel vises medarbejdernavne, som har syv bogstaver og starter med 'S':

```
SELECT NAME
   FROM STAFF
   WHERE NAME LIKE 'S _ _ _ _ _ _'
```

I det næste eksempel vises navnene på medarbejdere, der ikke starter med bogstavet 'S':

```
SELECT NAME
   FROM STAFF
   WHERE NAME  NOT LIKE 'S%'
```

Prædikatet EXISTS

Du kan bruge en underforespørgsel til at kontrollere, om der *eksisterer* en række, som opfylder en given betingelse. I det tilfælde kædes underforespørgslen sammen med den ydre forespørgsel vha. prædikatet EXISTS eller NOT EXISTS.

Når du kæder en underforespørgsel sammen med en ydre forespørgsel vha. prædikatet EXISTS, returneres der ikke en værdi. Prædikatet EXISTS er sandt, hvis resultatet af underforespørgslen indeholder én eller flere rækker, og falsk, hvis det ikke indeholder nogen.

Prædikatet EXISTS bruges tit sammen med korrelerede underforespørgsler. I eksemplet nedenfor vises de afdelinger, som pt. ikke har nogen registrering i tabellen PROJECT:

```
SELECT DEPTNO, DEPTNAME
   FROM DEPARTMENT X
   WHERE NOT EXISTS
```

```
            (SELECT *
                FROM PROJECT
                WHERE DEPTNO = X.DEPTNO)
    ORDER BY DEPTNO
```

Prædikaterne EXISTS og NOT EXISTS kan forbindes med andre prædikater vha. AND og OR i WHERE-udtrykket i den ydre forespørgsel.

Kvantificerede prædikater

Et kvantificeret prædikat sammenligner en værdi med en gruppe af værdier. Hvis en fullselect returnerer mere end én værdi, skal du ændre sammenligningsoperatorerne i prædikatet ved at tilføje ALL, ANY eller SOME. De angiver, hvordan de returnerede værdier skal behandles i det ydre prædikat. Her vises en række eksempler med sammenligningsoperatoren >, men bemærkningerne nedenfor gælder også for de andre operatorer:

udsagn > ALL (fullselect)
Prædikatet er sandt, hvis udtrykket er større end hver enkelt værdi, som fullselecten returnerer. Hvis fullselecten ikke returnerer nogen værdier, er prædikatet sandt. Hvis resultatet er falsk, er det angivne forhold falsk for mindst én værdi. Bemærk, at det kvantificerede prædikat <>ALL svarer til prædikatet NOT IN.

I det følgende eksempel bruges en underforespørgsel og sammenligningen > ALL til at finde navn og stilling for alle medarbejdere, som tjener mere end alle ledere:

```
SELECT LASTNAME, JOB
   FROM EMPLOYEE
   WHERE SALARY > ALL
   (SELECT SALARY
       FROM EMPLOYEE
       WHERE JOB='MANAGER')
```

udsagn > ANY (fullselect)
Prædikatet er sandt, hvis udtrykket er større end mindst én af de værdier, som fullselecten returnerer. Hvis fullselecten ikke returnerer nogen værdier, er prædikatet falsk. Bemærk, at det kvantificerede prædikat =ANY svarer til prædikatet IN.

udsagn > SOME (fullselect)
SOME er synonymt med ANY.

Der er flere oplysninger om prædikater og operatorer i *SQL Reference*.

Kapitel 7. Udvidet SQL

Dette kapitel dækker flere funktioner i DB2 Universal Database, som giver dig mulighed for at udforme forespørgsler mere effektivt og tilpasse dem til dine behov. Emnerne i kapitlet forudsætter, at du har forstået det, der hidtil er gennemgået.

Der er følgende afsnit i kapitlet:

- Anvend betingelser og triggere til forretningsregler
- Sammenkædninger
- ROLLUP- og CUBE-forespørgsler og Rekursive forespørgsler
- OLAP-funktioner

Anvend betingelser og triggere til forretningsregler

I erhvervslivet er der visse regler, der altid skal overholdes. En medarbejder, der arbejder på et projekt, skal f.eks. stå på lønningslisten. Der kan også være bestemte aktiviteter, der skal udføres automatisk. Når en sælger f.eks. får en ordre i hus, skal hans eller hendes provision stige.

DB2 Universal Database har en række nyttige funktioner til dette formål:

- En *entydig betingelse* medfører, at én eller flere kolonner i en tabel ikke kan indeholde to ens værdier.
- En *referenceintegritetsbetingelse* sørger for, at data er konsistente i alle de angivne tabeller.
- *Tabelkontrolbetingelser* er regler, som begrænser de værdier, en kolonne må indeholde. Data kan ikke indsættes eller opdateres i en kolonne, hvis værdien ikke opfylder kontrolbetingelserne for kolonnen.
- En *trigger* definerer en række handlinger, som udføres (udløses) efter en sletning, indsættelse eller opdatering i en given tabel. Triggere kan bruges til at skrive til andre tabeller, ændre inputværdier og afsende varsler.

Det første afsnit indeholder en oversigt over nøgler. Senere gennemgås referenceintegritet, betingelser og triggere vha. eksempler og diagrammer.

Nøgler

En *nøgle* er et sæt kolonner, som kan bruges til at identificere eller få adgang til en bestemt række eller bestemte rækker.

En nøgle, der består af mere end én kolonne, kaldes en *sammensat nøgle*. I en tabel med en sammensat nøgle svarer rækkefølgen af kolonnerne inden for den sammensatte nøgle ikke nødvendigvis til deres rækkefølge i tabellen.

Entydige nøgler

En *entydig nøgle* er defineret som en eller flere kolonner, hvor hver kolonne ikke må indeholde to ens værdier. Kolonnerne i en entydig nøgle må ikke indeholde NULL-værdier. Når INSERT- og UPDATE-sætninger udføres, kontrollerer databasesystemet, at betingelsen overholdes. En tabel kan have flere entydige nøgler. Entydige nøgler er valgfri og kan defineres i CREATE TABLE- eller ALTER TABLE-sætninger.

Primærnøgler

En *primærnøgle* er en entydig nøgle, som er en del af tabeldefinitionen. En tabel kan kun have én primærnøgle, og primærnøglen må ikke indeholde NULL-værdier. Primærnøgler er valgfri og kan defineres i CREATE TABLE- eller ALTER TABLE-sætninger.

Fremmednøgler

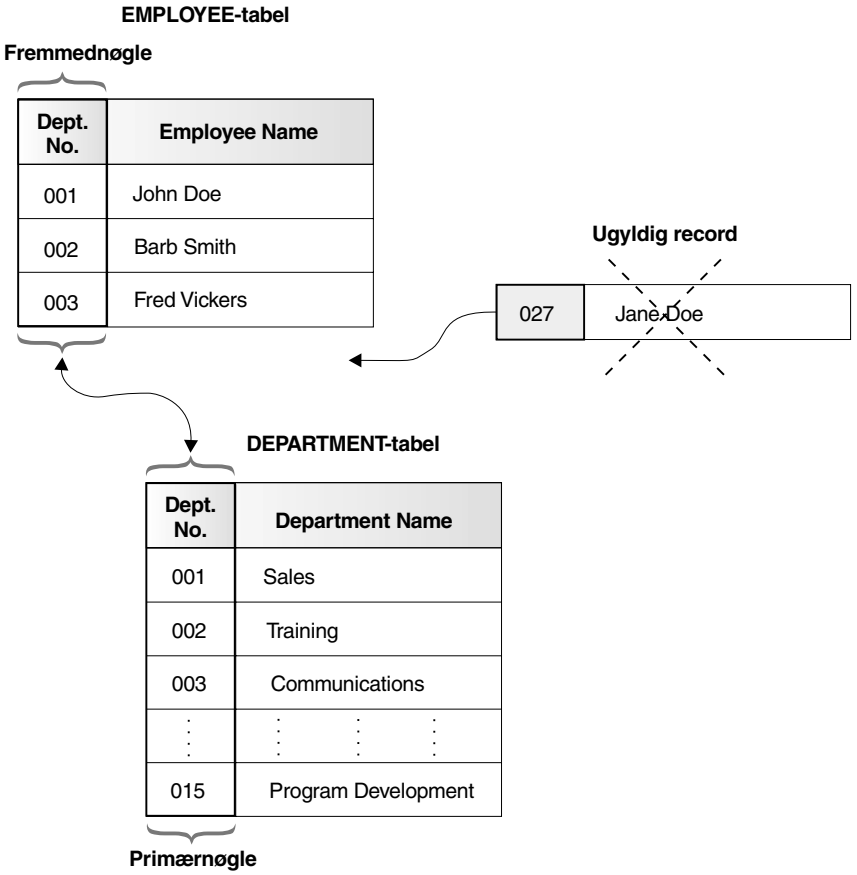
En *fremmednøgle* angives i definitionen af en referencebetingelse. En tabel kan have nul eller flere fremmednøgler. Værdien for den sammensatte fremmednøgle er NULL, hvis en komponent i værdien er NULL. Fremmednøgler er valgfri og kan defineres i CREATE TABLE- eller ALTER TABLE-sætninger.

Entydige betingelser

En entydig betingelse sørger for, at værdierne for en nøgle en entydige inden for en tabel. Entydige betingelser er valgfri. Du kan definere dem vha. sætningen CREATE TABLE eller ALTER TABLE ved at angive udtrykket PRIMARY KEY eller UNIQUE. For eksempel kan du definere en entydig betingelse for en tabelkolonne, der indeholder personalenummer, så alle medarbejdere får et entydigt nummer.

Referenceintegritetsbetingelser

Vha. entydige betingelser og fremmednøgler kan du angive koblinger mellem tabeller og dermed sørge for, at visse forretningsregler overholdes. Kombinationen entydig nøgle og fremmednøglebetingelser kaldes ofte referenceintegritetsbetingelser. En entydig betingelse, som en fremmednøgle henviser til, kaldes en *overordnet nøgle*. En fremmednøgle henviser til eller er beslægtet med en bestemt overordnet nøgle. Der kan f.eks. være en regel, som siger, at alle medarbejdere (tabellen EMPLOYEE), skal tilhøre en eksisterende afdeling (tabellen DEPARTMENT). Derfor defineres afdelingsnummeret i tabellen EMPLOYEE som fremmednøgle, og afdelingsnummeret i tabellen DEPARTMENT som primærnøgle. Diagrammet nedenfor illustrerer, hvordan en referenceintegritetsbetingelse fungerer.



Figur 4. Fremmed- og primærnøglebetingelser definerer koblinger og beskytter data

Tabelkontrolbetingelser

En *tabelkontrolbetingelse* angiver en betingelse, som vurderes for hver række i en tabel. Du kan angive kontrolbetingelser for enkeltkolonner. Tilføj dem vha. sætningen CREATE TABLE eller ALTER TABLE.

Nedenfor oprettes en tabel med følgende betingelser:

- Afdelingsnummeret skal ligge mellem 10 og 100.
- En medarbejder skal have én af følgende stillinger: “Sales”, “Mgr” eller “Clerk”.
- Alle medarbejdere ansat før 1986 skal tjene over 40.500.

```
CREATE TABLE EMP
      (ID           SMALLINT NOT NULL,
       NAME         VARCHAR(9),
       DEPT         SMALLINT CHECK (DEPT BETWEEN 10 AND 100),
       JOB          CHAR(5)   CHECK (JOB IN ('Sales', 'Mgr', 'Clerk')),
       HIREDATE     DATE,
       SALARY       DECIMAL(7,2),
       COMM         DECIMAL(7,2),
       PRIMARY KEY (ID),
       CONSTRAINT YEARSAL CHECK
                          (YEAR(HIREDATE) >= 1986 OR SALARY > 40500) )
```

En betingelse overholdes, medmindre den er falsk. Hvis f.eks. DEPT er NULL for en indsat række, indsættes den uden fejl, selv om værdien for DEPT skal ligge mellem 10 og 100, som angivet i betingelsen.

I dette eksempel får tabellen EMPLOYEE tilføjet betingelsen COMP, som angiver at en medarbejders samlede aflønning skal overstige 15.000:

```
ALTER TABLE EMP
   ADD CONSTRAINT COMP CHECK (SALARY + COMM > 15000)
```

De eksisterende rækker i tabellen kontrolleres for at sikre, at de ikke er i strid med den nye betingelse. Du kan udskyde kontrollen ved at bruge SET CONSTRAINTS-sætningen sådan:

```
SET CONSTRAINTS FOR EMP OFF
ALTER TABLE EMP ADD CONSTRAINT COMP CHECK (SALARY + COMM > 15000)
SET CONSTRAINTS FOR EMP IMMEDIATE CHECKED
```

Først bruges SET CONSTRAINTS-sætningen til at udskyde betingelseskontrol for tabellen. Derefter kan én eller flere betingelser tilføjes, uden at tabellen kontrolleres. Derefter afsendes SET CONSTRAINTS-sætningen igen, så betingelseskontrol aktiveres, og udskudt kontrol foretages.

Triggere

En *trigger* definerer et sæt handlinger. Triggere aktiveres af funktioner, der ændrer data i en angivet basistabel.

Triggere kan f.eks. anvendes til:

- at validere inputdata
- automatisk at generere en værdi til en ny række
- at læse fra andre tabeller med henblik på krydsreferencer
- at skrive til andre tabeller med henblik på revisionsspor
- at sende varsler via elektronisk post

Med triggere bliver applikationsudvikling hurtigere, forretningsregler kan anvendes globalt, og det bliver lettere at vedligeholde applikationer og data.

DB2 Universal Database giver dig mulighed for at benytte flere typer triggere. Triggere kan defineres til at blive aktiveret før eller efter en DELETE-, INSERT- eller UPDATE-funktion. Hver trigger indeholder en række SQL-sætninger, som kaldes en *udløst handling*. Den kan bl.a. omfatte et valgfrit søgekriterium.

En *efter-trigger* (AFTER)kan defineres, så den udløste handling udføres for hver række, eller én gang for sætningen. *Før-triggere* (BEFORE) udfører altid den udløste handling for hver række.

Brug en trigger før en INSERT-, UPDATE- eller DELETE-sætning for at kontrollere visse betingelser, før en triggerhandling udføres, eller for at ændre inputværdierne, inden de gemmes i tabellen.

Brug en AFTER-trigger til at sprede værdier efter behov eller til at udføre andre funktioner, f.eks. afsendelse af en meddelelse, som kræves i forbindelse med triggerhandlingen.

I dette eksempel illustreres brugen af BEFORE- og AFTER-triggere. En applikation registrerer og sporer ændringer i aktiekurser. Databasen indeholder to tabeller, CURRENTQUOTE og QUOTEHISTORY, defineret som:

```
CREATE TABLE CURRENTQUOTE
(SYMBOL VARCHAR(10),
 QUOTE DECIMAL(5,2),
 STATUS VARCHAR(9))

CREATE TABLE  QUOTEHISTORY
(SYMBOL VARCHAR(10),
 QUOTE DECIMAL(5,2),
 TIMESTAMP TIMESTAMP)
```

Når kolonnen QUOTE i CURRENTQUOTE opdateres vha. en sætning som:

```
UPDATE CURRENTQUOTE
   SET QUOTE = 68.5
   WHERE SYMBOL = 'IBM'
```

skal kolonnen STATUS i CURRENTQUOTE opdateres med angivelse af, om aktiekursen er:

- stigende
- årets højeste
- faldende
- årets laveste
- stabil.

Det gøres vha. følgende BEFORE-trigger:

```
1
   CREATE TRIGGER STOCK_STATUS
      NO CASCADE BEFORE UPDATE OF QUOTE ON CURRENTQUOTE
      REFERENCING NEW AS NEWQUOTE OLD AS OLDQUOTE
      FOR EACH ROW MODE DB2SQL

2
    SET NEWQUOTE.STATUS =

3
       CASE

4
          WHEN NEWQUOTE.QUOTE >=
                    (SELECT MAX(QUOTE)
                        FROM QUOTEHISTORY
                        WHERE SYMBOL = NEWQUOTE.SYMBOL
                          AND YEAR(TIMESTAMP) = YEAR(CURRENT DATE) )
          THEN 'High'

5
        WHEN NEWQUOTE.QUOTE <=
                    (SELECT MIN(QUOTE)
                        FROM QUOTEHISTORY
                        WHERE SYMBOL = NEWQUOTE.SYMBOL
                        AND YEAR(TIMESTAMP) = YEAR(CURRENT DATE) )
        THEN 'Low'

6
        WHEN NEWQUOTE.QUOTE > OLDQUOTE.QUOTE
           THEN 'Rising'
        WHEN NEWQUOTE.QUOTE < OLDQUOTE.QUOTE
           THEN 'Dropping'
        WHEN NEWQUOTE.QUOTE = OLDQUOTE.QUOTE
           THEN 'Steady'
     END
```

1 Med denne kodeblok defineres STOCK_STATUS som en trigger, der skal aktiveres, før kolonnen QUOTE i tabellen CURRENTQUOTE opdateres. I anden linie angives det, at den udløste handling skal aktiveres før evt. ændringer, som den egentlige opdatering af tabellen CURRENTQUOTE medfører, aktiveres for databasen. Udtrykket NO CASCADE betyder, at den udløste handling ikke aktiverer andre triggere. I tredje linie angives de navne, som skal bruges som kvalifikatorer for kolonnenavnet for de nye værdier (NEWQUOTE) og de gamle værdier (OLDQUOTE). Kolonnenavne, der kvalificeres med disse kor-

relationsnavne (NEWQUOTE og OLDQUOTE), kaldes *overgangsvariabler*. Den fjerde linie angiver, at den udløste handling skal udføres for hver række.

2 Dette angiver begyndelsen på den første og eneste SQL-sætning i den udløste handling for triggeren. Sætningen SET overgangsvariabel bruges i en trigger til at knytte en værdi til en kolonne i rækken i den tabel, som opdateres vha. den sætning, som har aktiveret triggeren. Denne sætning knytter en værdi til kolonnen STATUS i tabellen CURRENTQUOTE.

3 Udtrykket til højre for tildelingen er et CASE-udtryk. CASE-udtrykket slutter ved nøgleordet END.

4 Først undersøges det, om den nye kurs (NEWQUOTE.QUOTE) er højere end eller lig med den højeste værdi for aktiesymbolet i indeværende kalenderår. Underforespørgslen bruger tabellen QUOTEHISTORY, som opdateres af den efterfølgende AFTER-trigger.

5 Først undersøges det, om den nye kurs (NEWQUOTE.QUOTE) er lavere end eller lig med den laveste værdi for aktiesymbolet i indeværende kalenderår. Underforespørgslen bruger tabellen QUOTEHISTORY, som opdateres af den efterfølgende AFTER-trigger.

6 I de sidste tre tilfælde sammenlignes den nye kurs (NEWQUOTE.QUOTE) med den hidtidige kurs (OLDQUOTE.QUOTE) for at afgøre, om den er højere, lavere eller uændret. Her slutter sætningen SET overgangsvariabel.

Når registreringen i tabellen CURRENTQUOTE opdateres, skal der også oprettes en kontrolrecord, ved at den nye kurs og dens tidsstempel kopieres til tabellen QUOTEHISTORY. Det gøres med følgende AFTER-trigger:

1

```
CREATE TRIGGER RECORD_HISTORY
AFTER UPDATE OF QUOTE ON CURRENTQUOTE
REFERENCING NEW AS NEWQUOTE
FOR EACH ROW MODE DB2SQL
BEGIN ATOMIC
```

2

```
INSERT INTO QUOTEHISTORY
   VALUES (NEWQUOTE.SYMBOL, NEWQUOTE.QUOTE, CURRENT TIMESTAMP);
END
```

1 Med denne kodeblok defineres en trigger, RECORD_HISTORY, der skal aktiveres, efter kolonnen QUOTE i tabellen CURRENTQUOTE opdateres. I tredje linie angives det navn, som skal bruges som kvalifikator for kolonnenavnet for den nye værdi (NEWQUOTE). Den fjerde linie angiver, at den udløste handling skal udføres for hver række.

2 Den udløste handling for denne trigger indeholder en enkelt SQL-sætning, som indsætter en række i tabellen QUOTEHISTORY vha. data fra den række, der er blevet opdateret (NEWQUOTE.SYMBOL og NEWQUOTE.QUOTE) og det aktuelle tidsstempel.

CURRENT TIMESTAMP er et specialregister, der indeholder tidsstemplet. Du kan finde en oversigt og forklaring under "Specialregistre" på side 68.

Sammenkædninger

Det at kombinere data fra to eller flere tabeller kaldes sammenkædning (JOIN). Databasesystemet opretter alle kombinationer af rækker fra de angivne tabeller. For hver kombination afprøves *sammenkædningsbetingelsen*. En sammenkædningsbetingelse er et søgekriterium med visse begrænsninger. *SQL Reference* indeholder en oversigt over begrænsningerne.

Bemærk, at datatyperne for de kolonner, sammenkædningsbetingelsen omfatter, ikke nødvendigvis skal være ens, men de skal være kompatible. Sammenkædningsbetingelsen vurderes på samme måde som andre søgekriterier, og de samme regler gælder for sammenligningen.

Hvis du ikke angiver en sammenkædningsbetingelse, vises alle kombinationer af rækker fra de tabeller, der angives i FROM-udtrykket, selv om rækkerne måske ikke har spor med hinanden at gøre. Resultatet kaldes *vektorproduktet* af de to tabeller.

Eksemplerne i dette afsnit er baseret på nedenstående to tabeller. Der er tale om forenklede udgaver af tabeller fra SAMPLE-databasen. De bruges til at illustrere et par generelle aspekter ved sammenkædning. SAMP_STAFF viser navnet på de medarbejdere, der ikke er kontraktansat, og deres jobbeskrivelser, mens SAMP_PROJECT viser navnet på medarbejderne (kontraktansatte og fastansatte) og de projekter, de arbejder med.

Tabellerne ser sådan ud:

NAME	PROJ
Haas	AD3100
Thompson	PL2100
Walker	MA2112
Lutz	MA2111

Figur 5. Tabellen SAMP_PROJECT

NAME	JOB
Haas	PRES
Thompson	MANAGER
Lucchessi	SALESREP
Nicholls	ANALYST

Figur 6. Tabellen SAMP_STAFF

I dette eksempel vises vektorproduktet af de to tabeller. Der er ikke angivet nogen sammenkædningsbetingelse, så alle kombinationer af rækker vises:

```
SELECT SAMP_PROJECT.NAME,
       SAMP_PROJECT.PROJ, SAMP_STAFF.NAME, SAMP_STAFF.JOB
  FROM SAMP_PROJECT, SAMP_STAFF
```

Resultatet er:

```
NAME       PROJ   NAME       JOB
---------- ------ ---------- --------
Haas       AD3100 Haas       PRES
Thompson   PL2100 Haas       PRES
Walker     MA2112 Haas       PRES
Lutz       MA2111 Haas       PRES
Haas       AD3100 Thompson   MANAGER
Thompson   PL2100 Thompson   MANAGER
Walker     MA2112 Thompson   MANAGER
Lutz       MA2111 Thompson   MANAGER
Haas       AD3100 Lucchessi  SALESREP
Thompson   PL2100 Lucchessi  SALESREP
Walker     MA2112 Lucchessi  SALESREP
Lutz       MA2111 Lucchessi  SALESREP
Haas       AD3100 Nicholls   ANALYST
Thompson   PL2100 Nicholls   ANALYST
Walker     MA2112 Nicholls   ANALYST
Lutz       MA2111 Nicholls   ANALYST
```

De to hovedtyper af sammenkædning er *indre sammenkædning* og *ydre sammenkædning*. Hidtil har vi benyttet indre sammenkædning i alle eksemplerne. Med indre sammenkædning gemmes kun de rækker i vektorproduktet, som opfylder sammenkædningsbetingelsen. Hvis en række findes i den ene tabel, men ikke i den anden, tages oplysningerne ikke med i resultattabellen.

I dette eksempel vises den indre sammenkædning mellem de to tabeller. Den indre sammenkædning viser de fastansatte medarbejdere, som er knyttet til et projekt:

```
SELECT SAMP_PROJECT.NAME,
       SAMP_PROJECT.PROJ, SAMP_STAFF.NAME, SAMP_STAFF.JOB
  FROM SAMP_PROJECT, SAMP_STAFF
  WHERE SAMP_STAFF.NAME = SAMP_PROJECT.NAME
```

Du kan også angive den indre sammenkædning sådan:

```
SELECT SAMP_PROJECT.NAME,
       SAMP_PROJECT.PROJ, SAMP_STAFF.NAME, SAMP_STAFF.JOB
  FROM SAMP_PROJECT INNER JOIN SAMP_STAFF
    ON SAMP_STAFF.NAME = SAMP_PROJECT.NAME
```

Resultatet er:

```
NAME       PROJ   NAME       JOB
---------- ------ ---------- --------
Haas       AD3100 Haas       PRES
Thompson   PL2100 Thompson   MANAGER
```

Bemærk, at resultatet af den indre sammenkædning består af rækker med ens værdier i kolonnen NAME i den højre og venstre tabel. Både 'Haas' og 'Thompson' er med i tabellen SAMP_STAFF, som viser alle fastansatte medarbejdere, og i tabellen SAMP_PROJECT, som viser fastansatte og kontraktansatte medarbejdere, der er knyttet til et projekt.

En ydre sammenkædning er en sammenkædning af den indre sammenkædning og rækker fra den venstre tabel, den højre tabel eller begge, som ikke er med i den indre sammenkædning. Når du udfører en ydre sammenkædning af to tabeller, kan du valgfrit angive den ene tabel som den venstre og den anden som den højre. Der er tre typer ydre sammenkædning:

1. En ***venstre ydre sammenkædning*** omfatter den indre sammenkædning og de rækker fra den venstre tabel, som ikke er med i den indre sammenkædning.
2. En ***højre ydre sammenkædning*** omfatter den indre sammenkædning og de rækker fra den højre tabel, som ikke er med i den indre sammenkædning.
3. En ***fuldstændig ydre sammenkædning*** omfatter den indre sammenkædning og de rækker fra både den venstre og den højre tabel, som ikke er med i den indre sammenkædning.

Brug SELECT-sætningen til at angive de kolonner, der skal vises. I FROM-udtrykket skal du angive navnet på den første tabel efterfulgt af nøgleordene LEFT OUTER JOIN, RIGHT OUTER JOIN eller FULL OUTER JOIN. Derefter skal du angive den anden tabel efterfulgt af nøgleordet ON. Efter nøgleordet ON skal du angive sammenkædningsbetingelsen for at angive koblingen mellem tabellerne.

I nedenstående eksempel er SAMP_STAFF den højre tabel og SAMP_PROJECT den venstre. Med LEFT OUTER JOIN får du vist navn og projektnum-

mer for alle medarbejdere, fastansatte som kontraktansatte, (registreret i SAMP_PROJECT) og deres stillingsbetegnelse, hvis de er fastansatte (registreret i SAMP_STAFF):

```
SELECT SAMP_PROJECT.NAME, SAMP_PROJECT.PROJ,
       SAMP_STAFF.NAME, SAMP_STAFF.JOB
  FROM SAMP_PROJECT LEFT OUTER JOIN SAMP_STAFF
    ON SAMP_STAFF.NAME = SAMP_PROJECT.NAME
```

Resultatet er:

```
NAME       PROJ                NAME       JOB
---------- ------------------- ---------- --------------------
Haas       AD3100              Haas       PRES
Lutz       MA2111              -          -
Thompson   PL2100              Thompson   MANAGER
Walker     MA2112              -          -
```

De rækker, hvor der er værdier i alle kolonner, er resultatet af den indre sammenkædning. Disse rækker opfylder sammenkædningsbetingelsen: 'Haas' og 'Thompson' er med i både SAMP_PROJECT (venstre tabel) og SAMP_STAFF (højre tabel). I de rækker, hvor sammenkædningsbetingelsen ikke opfyldes, vises NULL-værdien i kolonnerne for den højre tabel: 'Lutz' og 'Walker' er kontraktansatte og er med i tabellen SAMP_PROJECT men ikke i tabellen SAMP_STAFF. Bemærk, at alle rækker i den venstre tabel er med i resultatrækkerne.

I det næste eksempel er SAMP_STAFF den højre tabel og SAMP_PROJECT den venstre. Med RIGHT OUTER JOIN får du vist navn og stillingsbetegnelse for alle fastansatte medarbejdere (registreret i SAMP_STAFF) og deres projektnummer, hvis de har ét (registreret i SAMP_PROJECT):

```
SELECT SAMP_PROJECT.NAME,
       SAMP_PROJECT.PROJ, SAMP_STAFF.NAME, SAMP_STAFF.JOB
  FROM SAMP_PROJECT RIGHT OUTER JOIN SAMP_STAFF
    ON SAMP_STAFF.NAME = SAMP_PROJECT.NAME
```

Resultatet er:

```
NAME       PROJ                NAME       JOB
---------- ------------------- ---------- --------------------
Haas       AD3100              Haas       PRES
-          -                   Lucchessi  SALESREP
-          -                   Nicholls   ANALYST
Thompson   PL2100              Thompson   MANAGER
```

Som ved den venstre ydre sammenkædning er de rækker, hvor der er værdier i alle kolonner, resultatet af den indre sammenkædning. Disse rækker opfylder sammenkædningsbetingelsen: 'Haas' og 'Thompson' er med i både SAMP_PROJECT (venstre tabel) og SAMP_STAFF (højre tabel). I de rækker, hvor sammenkædningsbetingelsen ikke opfyldes, vises NULL-værdien i kolonnerne for den venstre tabel: 'Lucchessi' og 'Nicholls' er fastansatte medarbejdere,

som ikke er knyttet til et projekt. De er registreret i SAMP_STAFF, men ikke i SAMP_PROJECT. Bemærk, at alle rækker i den højre tabel er med i resultatrækkerne.

I det næste eksempel benyttes FULL OUTER JOIN sammen med tabellerne SAMP_PROJECT og SAMP_STAFF. Alle fastansatte medarbejdere, også dem der ikke er knyttet til et projekt, og alle kontraktansatte medarbejdere vises:

```
SELECT SAMP_PROJECT.NAME, SAMP_PROJECT.PROJ,
       SAMP_STAFF.NAME, SAMP_STAFF.JOB
  FROM SAMP_PROJECT FULL OUTER JOIN SAMP_STAFF
    ON SAMP_STAFF.NAME = SAMP_PROJECT.NAME
```

Resultatet er:

```
NAME       PROJ                NAME       JOB
---------- ------------------- ---------- --------------------
Haas       AD3100              Haas       PRES
-          -                   Lucchessi  SALESREP
-          -                   Nicholls   ANALYST
Thompson   PL2100              Thompson   MANAGER
Lutz       MA2111              -          -
Walker     MA2112              -          -
```

Resultatet omfatter den venstre ydre sammenkædning, den højre ydre sammenkædning og den indre sammenkædning. Alle fastansatte og kontraktansatte medarbejdere vises. Som ved den venstre og højre ydre sammenkædning indsættes der NULL-værdier, hvis sammenkædningsbetingelsen ikke er opfyldt. Alle rækker i SAMP_STAFF og SAMP_PROJECT vises i resultatrækkerne.

Komplekse forespørgsler

Med DB2 Universal Database kan du gruppere, sammentælle og få vist flere kolonner i ét sæt resultatrækker vha. ROLLUP og CUBE. Denne nye funktion forbedrer og forenkler dataanalyse vha. SQL.

Du kan hente oplysninger i databasen på flere måder. Med rekursive forespørgsler kan du få vist resultattabeller fra eksisterende datasæt.

ROLLUP- og CUBE-forespørgsler

Du kan angive ROLLUP- og CUBE-funktioner i GROUP BY-udtrykket i en forespørgsel.

ROLLUP-gruppering giver resultatrækker, der indeholder de almindelige grupperede rækker samt *kolonnesammentællinger*. CUBE-gruppering giver resultatrækker, der indeholder rækkerne fra ROLLUP samt rækkesammentællinger.

Med ROLLUP kunne du f.eks. få vist salg pr. person pr. måned samt månedstotaler og den samlede total. Med CUBE får du yderligere vist rækker med samlet salg pr. person.

Der er flere oplysninger i *SQL Reference*.

Rekursive forespørgsler

En *rekursiv forespørgsel* er en forespørgsel, som iterativt bruger resultater til at finde frem til flere resultater. Det kan opfattes som en gennemgang af en træstruktur eller et diagram.

Eksempler fra virkeligheden er reservationssystemer og netværksplanlægning.

En rekursiv forespørgsel skrives som et fælles tabeludtryk, som indeholder en reference til sig selv.

Der er eksempler på rekursive forespørgsler i *SQL Reference*.

OLAP-funktioner

OLAP-funktioner (OnLine Analytical Processing) udfører beregningsfunktioner over et *vindue* af data. Vinduet kan angive en inddeling af rækker, en opstilling af rækkerne i inddelingerne eller en *beregningsgruppe*. Vha. beregningsfunktionen kan brugeren angive, hvilke rækker i forhold til den aktuell række der skal indgå i beregningen. Brugen af et sådan vindue gør det f.eks. muligt at udregne akkumuleret sum og bevægeligt gennemsnit.

Ud over de eksisterende beregningsfunktioner som SUM og AVG kan OLAP-funktionerne opstille elementer på en liste (RANK og DENSE_RANK) og nummerere rækker (ROW_NUMBER) ud fra en bestemt inddeling og opstilling af rækker.

I forespørgslen nedenfor vises medarbejdernes placering på lønningslisten og den samlede lønsum i afdeling 15 og 38:

```
SELECT NAME, DEPT,
       RANK () OVER (PARTITION BY DEPT ORDER BY SALARY DESC) AS RANK,
       SUM (SALARY) OVER (PARTITION BY DEPT
                          ORDER BY SALARY DESC
                          ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
           AS CUMULATIVE_SUM
  FROM STAFF
  WHERE DEPT IN (15,38)
  ORDER BY DEPT, RANK
```

Resultatet er:

```
NAME       DEPT      RANK       CUMULATIVE_SUM
---------- --------- ---------- --------------------
Hanes      15                 1             20659.80
Rothman    15                 2             37162.63
Ngan       15                 3             49670.83
Kermisch   15                 4             61929.33
O'Brien    38                 1             18006.00
Marenghi   38                 2             35512.75
Quigley    38                 3             52321.05
Naughton   38                 4             65275.80
Abrahams   38                 5             77285.55
```

Kapitel 8. Tilpasning og forbedring af datamanipulation

Dette kapitel indeholder en kort introduktion til *objektorienterede udvidelser* i DB2 Universal Database. Der er mange fordele ved at bruge objektorienterede udvidelser. *Brugerdefinerede typer* (UDT) gør flere datatyper tilgængelige for applikationerne, mens *brugerdefinerede funktioner* (UDF) gør det muligt at oprette applikationsspecifikke funktioner. Brugerdefinerede funktioner virker som *metoder* for brugerdefinerede typer ved at sørge for konsistente reaktioner og indkapsle typerne.

Derefter behandles *specialregistre* og *systemkataloger*. Specialregistre er lagerområder, der defineres af databasesystemet. De bruges til at opbevare oplysninger, som SQL-sætninger anvender. Specialregistre oprettes til brug for en bestemt applikation, når den opretter forbindelse til databasen. Systemkatalogerne indeholder oplysninger om databaseobjekternes logiske og fysiske struktur.

Der er følgende afsnit i kapitlet:

- Brugerdefinerede typer
- Brugerdefinerede funktioner
- Store objekter (LOB)
- Specialregistre
- Introduktion til katalogudpluk

En grundigere gennemgang af emnerne ligger uden for rammerne af denne bog. Der henvises til *SQL Reference* og *Administration Guide*.

Brugerdefinerede typer

En *DISTINCT-type* er en brugerdefineret type, som har samme interne repræsentation som en eksisterende type (dens "kildetype") men betragtes som selvstændig og ikke-kompatibel i forbindelse med de fleste funktioner. Du kan f.eks. definere typer for alder, vægt og højde, som jo alle er forskellige, men som bruger den indbyggede datatype INTEGER (heltal) til deres interne repræsentationer.

Følgende eksempel viser, hvordan en DISTINCT-type ved navn PAY oprettes:

```
CREATE DISTINCT TYPE PAY AS DECIMAL(9,2) WITH COMPARISONS
```

Selv om PAY har samme repræsentation som den indbyggede datatype DECIMAL(9,2), betragtes den som en selvstændig type, der ikke kan sammenlignes

med DECIMAL(9,2) eller nogen anden type. Den kan kun sammenlignes med samme DISTINCT-type. Operatorer og funktioner, som fungerer sammen med DECIMAL, kan heller ikke bruges her. Således kan en PAY-værdi ikke ganges med en INTEGER-værdi. Du skal derfor skrive funktioner, som kun gælder for datatypen PAY.

Hvis du bruger DISTINCT-datatyper, mindskes risikoen for fejltagelser. Hvis kolonnen SALARY i tabellen EMPLOYEE er defineret til datatypen PAY, kan den ikke lægges sammen med COMM, selv om de har samme kildetype.

DISTINCT-datatyper tillader konvertering. En kildetype kan konverteres til en DISTINCT-datatype og omvendt. Hvis kolonnen SALARY i tabellen EMPLOYEE er defineret til datatypen PAY, kan følgende eksempel udføres uden fejl af sammenligningsoperatoren.

```
SELECT * FROM EMPLOYEE
   WHERE DECIMAL(SALARY) = 41250
```

DECIMAL(SALARY) returnerer datatype DECIMAL. Omvendt kan en numerisk datatype konverteres til typen PAY. Du kan f.eks. konvertere tallet 41250 vha. PAY(41250).

Brugerdefinerede funktioner

Som nævnt under "Funktioner" på side 28, indeholder DB2 Universal Database indbyggede og brugerdefinerede funktioner. Disse funktioner kan imidlertid ikke dække alle behov. I mange tilfælde vil du få brug for at tilpasse funktioner til bestemte opgaver. Med brugerdefinerede funktioner kan du oprette sådanne funktioner.

Der er fire typer brugerdefinerede funktioner: *SOURCE* (kilde), *EXTERNAL SCALAR* (ekstern skalar), *EXTERNAL TABLE* (ekstern tabel) og *OLE DB EXTERNAL TABLE*.

Dette afsnit omhandler SOURCE- og EXTERNAL SCALAR-typer. Der er flere oplysninger om EXTERNAL TABLE- og OLE DB EXTERNAL TABLE-typer i *SQL Reference*.

Brugerdefinerede kildefunktioner gør det muligt for brugerdefinerede typer selektivt at henvise til en anden indbygget eller brugerdefineret funktion, som databasen allerede kender. Du kan bruge både skalare funktioner og beregningsfunktioner.

I det næste eksempel oprettes en brugerdefineret funktion (MAX), der er baseret på den indbyggede kolonnefunktion MAX, som benytter input med datatypen DECIMAL. Den brugerdefinerede funktion MAX benytter input med typen PAY og returnerer output med typen PAY.

```
CREATE FUNCTION MAX(PAY) RETURNS PAY
   SOURCE MAX(DECIMAL)
```

Eksterne brugerdefinerede funktioner skrives af brugere i et programmeringssprog. Der er *eksterne skalarfunktioner* og *eksterne tabelfunktioner*. Begge typer gennemgås i *SQL Reference*.

Hvis du f.eks. allerede har oprettet en funktion, som tæller antallet af ord i en streng, kan du registrere den i databasen vha. CREATE FUNCTION-sætningen og give den navnet WORDCOUNT. Funktionen kan så bruges i SQL-sætninger.

Nedenstående sætning returnerer f.eks. personalenumre og antal ord i medarbejdernes CV i ASCII-format. WORDCOUNT er en ekstern skalarfunktion, som brugeren har registreret i databasen, og som nu bruges i sætningen.

```
SELECT EMPNO, WORDCOUNT(RESUME)
   FROM EMP_RESUME
   WHERE RESUME_FORMAT = 'ascii'
```

Der er flere oplysninger om oprettelse af brugerdefinerede funktioner i *Application Development Guide*.

Store objekter (LOB)

Udtrykket *stort objekt* eller *LOB* bruges om tre datatyper: BLOB, CLOB og DBCLOB. Disse typer kan indeholde store mængder data til objekter som lyd, billeder og dokumenter.

Et *binært stort objekt (BLOB)* en en streng med variabel længde, målt i byte. Den kan blive op til 2 GB lang. Et BLOB bruges primært til specielle data som billeder, lyd og blandede medier.

Et *stort objekt af typen Character (CLOB)* er en streng med variabel længde, målt i byte. Den kan være op til 2 GB lang. Et CLOB bruges til at gemme enkeltbyte data, f.eks. dokumenter. Et CLOB betragtes som en tegnstreng.

Et *stort, dobbeltbyteobjekt af typen Character (DBCLOB)* er en streng af variabel længde, som indeholder dobbeltbytetegn. Det kan være op til 2 GB langt (1.073.741.823 dobbeltbytetegn). Et DBCLOB bruges til at gemme store dobbeltbytedata, f.eks. dokumenter. Et DBCLOB betragtes som en grafikstreng.

Håndtér store objekter (LOB)

Da LOB-værdier kan være meget store, kan det tage tid at overføre dem fra databaseserveren til klientapplikationen. Normalt behandles LOB-værdier dog

bid for bid, ikke samlet. I de tilfælde, hvor en applikation ikke behøver at gemme hele LOB-værdien i hukommelsen, kan den henvise til den med en *LOB-lokalisator*variabel.

Efterfølgende sætninger bruger lokalisatorerne til at udføre funktioner med dataene uden at skulle hente hele det store objekt. Lokalisatorvariabler bruges til at nedsætte kravene til applikationernes hukommelse og forbedre performance ved at reducere dataoverførslen mellem klientsystemet og serveren.

En anden mulighed er *filreferencevariabler*. De bruges til at hente et stort objekt direkte ind i en fil eller til at opdatere et stort objekt i en tabel direkte fra en fil. Filreferencevariabler bruges til at nedsætte kravene til applikationernes hukommelse, da det ikke er nødvendigt for applikationerne at gemme de store objekter. Der er flere oplysninger i *Application Development Guide* og *SQL Reference*.

Specialregistre

Et *specialregister* er et lageområde, der er defineret til en forbindelse af databasesystemet, og som bruges til at gemme oplysninger, der kan henvises til i SQL-sætninger. Her følger et par eksempler på almindelige specialregistre. Du kan finde en oversigt over alle specialregistre og en mere udførlig beskrivelse i *SQL Reference*.

- CURRENT DATE: Indeholder datoen ifølge systemuret ved udførelsen af SQL-sætningen.
- CURRENT FUNCTION PATH: Indeholder en værdi, som angiver den funktionssti, der bruges til at opløse funktions- og datatypehenvisninger.
- CURRENT SERVER: Angiver den aktuelle applikationsserver.
- CURRENT TIME: Indeholder klokkeslættet ifølge systemuret ved udførelsen af SQL-sætningen.
- CURRENT TIMESTAMP: Angiver et tidsstempel ifølge systemuret ved udførelsen af SQL-sætningen.
- CURRENT TIMEZONE: Angiver forskellen mellem Universal Time Coordinated og den lokale tid på applikationsserveren.
- USER: Angiver autorisations-id ved udførelse af programmet.

Du kan få vist indholdet af et specialregister med VALUES-sætningen. Eksempel:

```
VALUES (CURRENT TIMESTAMP)
```

Du kan også bruge:

```
SELECT CURRENT TIMESTAMP FROM ORG
```

Her får du vist tidsstemplet for hver række, der er registreret i tabellen.

Introduktion til katalogudpluk

DB2 opretter og vedligeholder en stor gruppe systemkatalogtabeller for hver enkelt database. Disse tabeller indeholder oplysninger om den logiske og fysiske struktur på databaseobjekter som f.eks. tabeller, udpluk, pakker, referenceintegritetsrelationer, funktioner, DISTINCT-typer og triggere. De oprettes, når databasen oprettes, og opdateres i løbet af den almindelige drift. Objekterne kan ikke eksplicit oprettes eller slettes, men du kan rette forespørgsler mod dem og få vist deres indhold.

Der er flere oplysninger i *SQL Reference*.

Vælg rækker fra systemkataloger

Katalogudpluk er som alle andre databaseudpluk. Med SQL-sætninger kan du få vist data på nøjagtig samme måde som i ethvert andet udpluk på systemet.

Du kan finde nyttige oplysninger om tabeller i kataloget SYSCAT.TABLES. Hvis du vil finde frem til navnene på eksisterende tabeller, som du har oprettet, skal du f.eks. skrive følgende:

```
SELECT TABNAME, TYPE, CREATE_TIME
   FROM SYSCAT.TABLES
   WHERE DEFINER = USER
```

Resultatet er:

```
TABNAME            TYPE CREATE_TIME
------------------ ---- --------------------------
ORG                T    1999-07-21-13.42.55.128005
STAFF              T    1999-07-21-13.42.55.609001
DEPARTMENT         T    1999-07-21-13.42.56.069001
EMPLOYEE           T    1999-07-21-13.42.56.310001
EMP_ACT            T    1999-07-21-13.42.56.710001
PROJECT            T    1999-07-21-13.42.57.051001
EMP_PHOTO          T    1999-07-21-13.42.57.361001
EMP_RESUME         T    1999-07-21-13.42.59.154001
SALES              T    1999-07-21-13.42.59.855001
CL_SCHED           T    1999-07-21-13.43.00.025002
IN_TRAY            T    1999-07-21-13.43.00.055001
```

Nedenstående oversigt indeholder katalogudpluk, som vedrører emner, der er gennemgået i denne bog. Der er mange flere katalogudpluk, som du kan læse om i bøgerne *SQL Reference* og *Administration Guide*.

Beskrivelse	Katalogudpluk
kontrolbetingelser	SYSCAT.CHECKS
kolonner	SYSCAT.COLUMNS
kolonner, som kontrolbetingelser henviser til	SYSCAT.COLCHECKS
kolonner, der bruges i nøgler	SYSCAT.KEYCOLUSE

Beskrivelse	Katalogudpluk
datatyper	SYSCAT.DATATYPES
funktionsparametre eller resultatet af en funktion	SYSCAT.FUNCPARMS
referencebetingelser	SYSCAT.REFERENCES
skemaer	SYSCAT.SCHEMATA
tabelbetingelser	SYSCAT.TABCONST
tabeller	SYSCAT.TABLES
triggere	SYSCAT.TRIGGERS
brugerdefinerede funktioner	SYSCAT.FUNCTIONS
udpluk	SYSCAT.VIEWS

Tillæg A. Tabeller i SAMPLE-databasen

I dette tillæg vises de oplysninger, der findes i tabeleksemplerne i eksempeldatabasen SAMPLE, og hvordan de oprettes og fjernes.

Der følger flere eksempeldatabaser med DB2 Universal Database. De viser funktioner i Business Intelligence og bruges i øvelserne i Business Intelligence. Det er kun indholdet af eksempeldatabasen SAMPLE, der vises i dette tillæg. Der er flere oplysninger om eksempeldatabaserne til Business Intelligence i *Data Warehouse Center Administration Guide*.

Tabeleksemplerne er dem, som bruges i eksemplerne i denne manual og andre manualer i programdokumentationen. Desuden vises de data, som findes i eksempelfilerne med data af typen BLOB og CLOB.

Tillægget indeholder følgende afsnit:

I tabeleksemplerne angiver en tankestreg (-) en NULL-værdi.

SAMPLE-databasen

Eksemplerne i bogen kommer fra en eksempeldatabase, som hedder SAMPLE. Hvis du vil bruge eksemplerne, skal du oprette SAMPLE-databasen. Det kræver, at du først har installeret databasesystemet.

Opret SAMPLE-databasen

SAMPLE-databasen oprettes vha. en fil, der kan eksekveres.[2] Du skal have SYSADM-autorisation for at kunne oprette en database.

- **UNIX-platforme**

  Hvis du bruger styresystemets kommandolinie, skal du skrive:

  ```
  sqllib/bin/db2sampl <sti>
  ```

  fra det personlige bibliotek for ejeren af databasesubsystemet, hvor *sti* er en valgfri parameter, der angiver den sti, hvor SAMPLE-databasen skal oprettes. Tryk på Enter.[3] Skemaet for DB2SAMPL er specialregisterværdien CURRENT SCHEMA.

- **OS/2- eller Windows-platforme**

  Hvis du bruger styresystemets kommandolinie, skal du skrive:

  ```
  db2sampl e
  ```

  hvor *e* er en valgfri parameter, der angiver det drev, hvor databasen skal oprettes. Tryk på Enter.[4]

  Hvis du ikke har logget på arbejdsstationen vha. Håndtering af brugerprofil, bliver du bedt om at gøre det.

Slet SAMPLE-databasen

Hvis du ikke har brug for at få adgang til SAMPLE-databasen, kan du slette den vha. kommandoen DROP DATABASE:

```
db2 drop database sample
```

Tabellen CL_SCHED

Navn:	CLASS_CODE	DAY	STARTING	ENDING
Type:	char(7)	smallint	time	time
Beskrivelse:	Timekode (lokale:lærer)	Dagnr. i 4-dages skema	Starttidspunkt for time	Sluttidspunkt for time

2. Der er flere oplysninger under kommandoen DB2SAMPL i *Command Reference*.

3. Hvis du ikke angiver nogen sti, oprettes tabeleksemplerne i den standardsti, som parameteren DFTDBPATH i konfigurationsfilen til databasesystemet angiver.

4. Hvis du ikke angiver noget drev, oprettes eksempeldatabasen på det samme drev som DB2.

Tabellen DEPARTMENT

Navn:	DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
Type:	char(3) not null	varchar(29) not null	char(6)	char(3) not null	char(16)
Beskrivelse:	Afdelingsnummer	Navn, der beskriver afdelingens hovedopgaver	Afdelingslederens personalenummer (EMPNO)	Afdeling (DEPTNO), som denne afdeling rapporterer til	Navn på ekstern lokation
Værdier:	A00	SPIFFY COMPUTER SERVICE DIV.	000010	A00	-
	B01	PLANNING	000020	A00	-
	C01	INFORMATION CENTER	000030	A00	-
	D01	DEVELOPMENT CENTER	-	A00	-
	D11	MANUFACTURING SYSTEMS	000060	D01	-
	D21	ADMINISTRATION SYSTEMS	000070	D01	-
	E01	SUPPORT SERVICES	000050	A00	-
	E11	OPERATIONS	000090	E01	-
	E21	SOFTWARE SUPPORT	000100	E01	-

Tabellen EMPLOYEE

Navn:	EMPNO	FIRSTNME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE
Type:	char(6) not null	varchar(12) not null	char(1) not null	varchar(15) not null	char(3)	char(4)	date
Beskrivelse:	Personalenummer	Fornavn	Initial for mellemnavn	Efternavn	Medarbejders afdeling (DEPTNO)	Telefonnummer	Ansættelsesdato

JOB	EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
char(8)	smallint not null	char(1)	date	dec(9,2)	dec(9,2)	dec(9,2)
Job	Uddannelsesvarighed i år	Køn (M mand, F kvinde)	Fødselsdato	Årsløn	Årsbonus	Årskommission

På næste side kan du se værdierne i tabellen EMPLOYEE.

EMPNO	FIRSTNME	MID INIT	LASTNAME	WORK DEPT	PHONE NO	HIREDATE	JOB	ED LEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
char(6) not null	varchar(12) not null	char(1) not null	varchar(15) not null	char(3)	char(4)	dato	char(8)	smal-lint not null	char(1)	date	dec(9,2)	dec(9,2)	dec(9,2)
000010	CHRISTINE	I	HAAS	A00	3978	1965-01-01	PRES	18	F	1933-08-24	52750	1000	4220
000020	MICHAEL	L	THOMPSON	B01	3476	1973-10-10	MANAGER	18	M	1948-02-02	41250	800	3300
000030	SALLY	A	KWAN	C01	4738	1975-04-05	MANAGER	20	F	1941-05-11	38250	800	3060
000050	JOHN	B	GEYER	E01	6789	1949-08-17	MANAGER	16	M	1925-09-15	40175	800	3214
000060	IRVING	F	STERN	D11	6423	1973-09-14	MANAGER	16	M	1945-07-07	32250	500	2580
000070	EVA	D	PULASKI	D21	7831	1980-09-30	MANAGER	16	F	1953-05-26	36170	700	2893
000090	EILEEN	W	HENDERSON	E11	5498	1970-08-15	MANAGER	16	F	1941-05-15	29750	600	2380
000100	THEODORE	Q	SPENSER	E21	0972	1980-06-19	MANAGER	14	M	1956-12-18	26150	500	2092
000110	VINCENZO	G	LUCCHESSI	A00	3490	1958-05-16	SALESREP	19	M	1929-11-05	46500	900	3720
000120	SEAN		O'CONNELL	A00	2167	1963-12-05	CLERK	14	M	1942-10-18	29250	600	2340
000130	DOLORES	M	QUINTANA	C01	4578	1971-07-28	ANALYST	16	F	1925-09-15	23800	500	1904
000140	HEATHER	A	NICHOLLS	C01	1793	1976-12-15	ANALYST	18	F	1946-01-19	28420	600	2274
000150	BRUCE		ADAMSON	D11	4510	1972-02-12	DESIGNER	16	M	1947-05-17	25280	500	2022
000160	ELIZABETH	P	PIANKA	D11	3782	1977-10-11	DESIGNER	17	F	1955-04-12	22250	400	1780
000170	MASATOSHI	J	YOSHIMURA	D11	2890	1978-09-15	DESIGNER	16	M	1951-01-05	24680	500	1974
000180	MARILYN	S	SCOUTTEN	D11	1682	1973-07-07	DESIGNER	17	F	1949-02-21	21340	500	1707
000190	JAMES	H	WALKER	D11	2986	1974-07-26	DESIGNER	16	M	1952-06-25	20450	400	1636
000200	DAVID		BROWN	D11	4501	1966-03-03	DESIGNER	16	M	1941-05-29	27740	600	2217
000210	WILLIAM	T	JONES	D11	0942	1979-04-11	DESIGNER	17	M	1953-02-23	18270	400	1462
000220	JENNIFER	K	LUTZ	D11	0672	1968-08-29	DESIGNER	18	F	1948-03-19	29840	600	2387
000230	JAMES	J	JEFFERSON	D21	2094	1966-11-21	CLERK	14	M	1935-05-30	22180	400	1774
000240	SALVATORE	M	MARINO	D21	3780	1979-12-05	CLERK	17	M	1954-03-31	28760	600	2301
000250	DANIEL	S	SMITH	D21	0961	1969-10-30	CLERK	15	M	1939-11-12	19180	400	1534
000260	SYBIL	P	JOHNSON	D21	8953	1975-09-11	CLERK	16	F	1936-10-05	17250	300	1380
000270	MARIA	L	PEREZ	D21	9001	1980-09-30	CLERK	15	F	1953-05-26	27380	500	2190
000280	ETHEL	P	SCHNEIDER	E11	8997	1967-03-24	OPERATOR	17	F	1936-03-28	26250	500	2100
000290	JOHN	P	PARKER	E11	4502	1980-05-30	OPERATOR	12	M	1946-07-09	15340	300	1227
000300	PHILIP	X	SMITH	E11	2095	1972-06-19	OPERATOR	14	M	1936-10-27	17750	400	1420
000310	MAUDE	F	SETRIGHT	E11	3332	1964-09-12	OPERATOR	12	F	1931-04-21	15900	300	1272

EMPNO	FIRSTNME	MID INIT	LASTNAME	WORK DEPT	PHONE NO	HIREDATE	JOB	ED LEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
000320	RAMLAL	V	MEHTA	E21	9990	1965-07-07	FIELDREP	16	M	1932-08-11	19950	400	1596
000330	WING		LEE	E21	2103	1976-02-23	FIELDREP	14	M	1941-07-18	25370	500	2030
000340	JASON	P	GOUNOT	E21	5698	1947-05-05	FIELDREP	16	M	1926-05-17	23840	500	1907

Tabellen EMP_ACT

Navn:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
Type:	char(6) not null	char(6) not null	smallint not null	dec(5,2)	date	date
Beskrivelse:	Personalenummer	Projektnummer	Aktivitetskode	Andel af medarbejders arbejdstid på projekt	Aktivitets startdato	Aktivitets slutdato
Værdier:	000010	AD3100	10	.50	1982-01-01	1982-07-01
	000070	AD3110	10	1.00	1982-01-01	1983-02-01
	000230	AD3111	60	1.00	1982-01-01	1982-03-15
	000230	AD3111	60	.50	1982-03-15	1982-04-15
	000230	AD3111	70	.50	1982-03-15	1982-10-15
	000230	AD3111	80	.50	1982-04-15	1982-10-15
	000230	AD3111	180	1.00	1982-10-15	1983-01-01
	000240	AD3111	70	1.00	1982-02-15	1982-09-15
	000240	AD3111	80	1.00	1982-09-15	1983-01-01
	000250	AD3112	60	1.00	1982-01-01	1982-02-01
	000250	AD3112	60	.50	1982-02-01	1982-03-15
	000250	AD3112	60	.50	1982-12-01	1983-01-01
	000250	AD3112	60	1.00	1983-01-01	1983-02-01
	000250	AD3112	70	.50	1982-02-01	1982-03-15
	000250	AD3112	70	1.00	1982-03-15	1982-08-15
	000250	AD3112	70	.25	1982-08-15	1982-10-15
	000250	AD3112	80	.25	1982-08-15	1982-10-15
	000250	AD3112	80	.50	1982-10-15	1982-12-01
	000250	AD3112	180	.50	1982-08-15	1983-01-01
	000260	AD3113	70	.50	1982-06-15	1982-07-01
	000260	AD3113	70	1.00	1982-07-01	1983-02-01
	000260	AD3113	80	1.00	1982-01-01	1982-03-01
	000260	AD3113	80	.50	1982-03-01	1982-04-15
	000260	AD3113	180	.50	1982-03-01	1982-04-15
	000260	AD3113	180	1.00	1982-04-15	1982-06-01
	000260	AD3113	180	.50	1982-06-01	1982-07-01
	000270	AD3113	60	.50	1982-03-01	1982-04-01
	000270	AD3113	60	1.00	1982-04-01	1982-09-01
	000270	AD3113	60	.25	1982-09-01	1982-10-15
	000270	AD3113	70	.75	1982-09-01	1982-10-15
	000270	AD3113	70	1.00	1982-10-15	1983-02-01

Navn:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
	000270	AD3113	80	1.00	1982-01-01	1982-03-01
	000270	AD3113	80	.50	1982-03-01	1982-04-01
	000030	IF1000	10	.50	1982-06-01	1983-01-01
	000130	IF1000	90	1.00	1982-01-01	1982-10-01
	000130	IF1000	100	.50	1982-10-01	1983-01-01
	000140	IF1000	90	.50	1982-10-01	1983-01-01
	000030	IF2000	10	.50	1982-01-01	1983-01-01
	000140	IF2000	100	1.00	1982-01-01	1982-03-01
	000140	IF2000	100	.50	1982-03-01	1982-07-01
	000140	IF2000	110	.50	1982-03-01	1982-07-01
	000140	IF2000	110	.50	1982-10-01	1983-01-01
	000010	MA2100	10	.50	1982-01-01	1982-11-01
	000110	MA2100	20	1.00	1982-01-01	1982-03-01
	000010	MA2110	10	1.00	1982-01-01	1983-02-01
	000200	MA2111	50	1.00	1982-01-01	1982-06-15
	000200	MA2111	60	1.00	1982-06-15	1983-02-01
	000220	MA2111	40	1.00	1982-01-01	1983-02-01
	000150	MA2112	60	1.00	1982-01-01	1982-07-15
	000150	MA2112	180	1.00	1982-07-15	1983-02-01
	000170	MA2112	60	1.00	1982-01-01	1983-06-01
	000170	MA2112	70	1.00	1982-06-01	1983-02-01
	000190	MA2112	70	1.00	1982-02-01	1982-10-01
	000190	MA2112	80	1.00	1982-10-01	1983-10-01
	000160	MA2113	60	1.00	1982-07-15	1983-02-01
	000170	MA2113	80	1.00	1982-01-01	1983-02-01
	000180	MA2113	70	1.00	1982-04-01	1982-06-15
	000210	MA2113	80	.50	1982-10-01	1983-02-01
	000210	MA2113	180	.50	1982-10-01	1983-02-01
	000050	OP1000	10	.25	1982-01-01	1983-02-01
	000090	OP1010	10	1.00	1982-01-01	1983-02-01
	000280	OP1010	130	1.00	1982-01-01	1983-02-01
	000290	OP1010	130	1.00	1982-01-01	1983-02-01
	000300	OP1010	130	1.00	1982-01-01	1983-02-01
	000310	OP1010	130	1.00	1982-01-01	1983-02-01
	000050	OP2010	10	.75	1982-01-01	1983-02-01
	000100	OP2010	10	1.00	1982-01-01	1983-02-01
	000320	OP2011	140	.75	1982-01-01	1983-02-01

Tabeller i SAMPLE-databasen

Navn:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
	000320	OP2011	150	.25	1982-01-01	1983-02-01
	000330	OP2012	140	.25	1982-01-01	1983-02-01
	000330	OP2012	160	.75	1982-01-01	1983-02-01
	000340	OP2013	140	.50	1982-01-01	1983-02-01
	000340	OP2013	170	.50	1982-01-01	1983-02-01
	000020	PL2100	30	1.00	1982-01-01	1982-09-15

Tabellen EMP_PHOTO

Navn:	EMPNO	PHOTO_FORMAT	PICTURE
Type:	char(6) not null	varchar(10) not null	blob(100k)
Beskrivelse:	Personalenummer	Fotoformat	Foto af medarbejder
Værdier:	000130	bitmap	db200130.bmp
	000130	gif	db200130.gif
	000130	xwd	db200130.xwd
	000140	bitmap	db200140.bmp
	000140	gif	db200140.gif
	000140	xwd	db200140.xwd
	000150	bitmap	db200150.bmp
	000150	gif	db200150.gif
	000150	xwd	db200150.xwd
	000190	bitmap	db200190.bmp
	000190	gif	db200190.gif
	000190	xwd	db200190.xwd

- "Billede af Quintana" på side 84 er et billede af medarbejderen Delores Quintana.
- "Billede af Nicholls" på side 86 er et billede af medarbejderen Heather Nicholls.
- "Billede af Adamson" på side 87 er et billede af medarbejderen Bruce Adamson.
- "Billede af Walker" på side 88 er et billede af medarbejderen James Walker.

Tabellen EMP_RESUME

Navn:	EMPNO	RESUME_FORMAT	RESUME
Type:	char(6) not null	varchar(10) not null	clob(5k)
Beskrivelse:	Personalenummer	CV-format	Medarbejder-CV
Værdier:	000130	ascii	db200130.asc
	000130	kommandofil	db200130.scr

Navn:	EMPNO	RESUME_FORMAT	RESUME
	000140	ascii	db200140.asc
	000140	kommandofil	db200140.scr
	000150	ascii	db200150.asc
	000150	kommandofil	db200150.scr
	000190	ascii	db200190.asc
	000190	kommandofil	db200190.scr

- Under "CV for Quintana" på side 85 finder du CV'et for medarbejderen Delores Quintana.
- Under "CV for Nicholls" på side 86 finder du CV'et for medarbejderen Heather Nicholls.
- Under "CV for Adamson" på side 87 finder du CV'et for medarbejderen Bruce Adamson.
- Under "CV for Walker" på side 89 finder du CV'et for medarbejderen James Walker.

Tabellen IN_TRAY

Navn:	RECEIVED	SOURCE	SUBJECT	NOTE_TEXT
Type:	timestamp	char(8)	char(64)	varchar(3000)
Beskrivelse:	Dato og klokkeslæt for modtagelse	Bruger-id på afsender	Kort beskrivelse	Meddelelsen

Tabellen ORG

Navn:	DEPTNUMB	DEPTNAME	MANAGER	DIVISION	LOCATION
Type:	smallint not null	varchar(14)	smallint	varchar(10)	varchar(13)
Beskrivelse:	Afdelingsnummer	Afdelingsnavn	Leders personalenummer	Selskabsdivision	By
Værdier:	10	Head Office	160	Corporate	New York
	15	New England	50	Eastern	Boston
	20	Mid Atlantic	10	Eastern	Washington
	38	South Atlantic	30	Eastern	Atlanta
	42	Great Lakes	100	Midwest	Chicago
	51	Plains	140	Midwest	Dallas
	66	Pacific	270	Western	San Francisco
	84	Mountain	290	Western	Denver

Tabellen PROJECT

Navn:	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
Type:	char(6) not null	varchar(24) not null	char(3) not null	char(6) not null	dec(5,2)	date	date	char(6)
Beskrivelse:	Projektnummer	Projektnavn	Ansvarlig afdeling	Ansvarlig medarbejder	Anslået gennemsnitsbemanding	Anslået startdato	Anslået slutdato	Hovedprojekt for underprojekt
Værdier:	AD3100	ADMIN SERVICES	D01	000010	6.5	1982-01-01	1983-02-01	-
	AD3110	GENERAL ADMIN SYSTEMS	D21	000070	6	1982-01-01	1983-02-01	AD3100
	AD3111	PAYROLL PROGRAMMING	D21	000230	2	1982-01-01	1983-02-01	AD3110
	AD3112	PERSONNEL PROGRAMMING	D21	000250	1	1982-01-01	1983-02-01	AD3110
	AD3113	ACCOUNT PROGRAMMING	D21	000270	2	1982-01-01	1983-02-01	AD3110
	IF1000	QUERY SERVICES	C01	000030	2	1982-01-01	1983-02-01	-
	IF2000	USER EDUCATION	C01	000030	1	1982-01-01	1983-02-01	-
	MA2100	WELD LINE AUTOMATION	D01	000010	12	1982-01-01	1983-02-01	-
	MA2110	W L PROGRAMMING	D11	000060	9	1982-01-01	1983-02-01	MA2100
	MA2111	W L PROGRAM DESIGN	D11	000220	2	1982-01-01	1982-12-01	MA2110
	MA2112	W L ROBOT DESIGN	D11	000150	3	1982-01-01	1982-12-01	MA2110
	MA2113	W L PROD CONT PROGS	D11	000160	3	1982-02-15	1982-12-01	MA2110
	OP1000	OPERATION SUPPORT	E01	000050	6	1982-01-01	1983-02-01	-
	OP1010	OPERATION	E11	000090	5	1982-01-01	1983-02-01	OP1000
	OP2000	GEN SYSTEMS SERVICES	E01	000050	5	1982-01-01	1983-02-01	-
	OP2010	SYSTEMS SUPPORT	E21	000100	4	1982-01-01	1983-02-01	OP2000
	OP2011	SCP SYSTEMS SUPPORT	E21	000320	1	1982-01-01	1983-02-01	OP2010
	OP2012	APPLICATIONS SUPPORT	E21	000330	1	1982-01-01	1983-02-01	OP2010

Navn:	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
	OP2013	DB/DC SUPPORT	E21	000340	1	1982-01-01	1983-02-01	OP2010
	PL2100	WELD LINE PLANNING	B01	000020	1	1982-01-01	1982-09-15	MA2100

Tabellen SALES

Navn:	SALES_DATE	SALES_PERSON	REGION	SALES
Type:	date	varchar(15)	varchar(15)	int
Beskrivelse:	Salgsdato	Medarbejders efternavn	Salgsområde	Antal salg
Værdier:	12/31/1995	LUCCHESSI	Ontario-South	1
	12/31/1995	LEE	Ontario-South	3
	12/31/1995	LEE	Quebec	1
	12/31/1995	LEE	Manitoba	2
	12/31/1995	GOUNOT	Quebec	1
	03/29/1996	LUCCHESSI	Ontario-South	3
	03/29/1996	LUCCHESSI	Quebec	1
	03/29/1996	LEE	Ontario-South	2
	03/29/1996	LEE	Ontario-North	2
	03/29/1996	LEE	Quebec	3
	03/29/1996	LEE	Manitoba	5
	03/29/1996	GOUNOT	Ontario-South	3
	03/29/1996	GOUNOT	Quebec	1
	03/29/1996	GOUNOT	Manitoba	7
	03/30/1996	LUCCHESSI	Ontario-South	1
	03/30/1996	LUCCHESSI	Quebec	2
	03/30/1996	LUCCHESSI	Manitoba	1
	03/30/1996	LEE	Ontario-South	7
	03/30/1996	LEE	Ontario-North	3
	03/30/1996	LEE	Quebec	7
	03/30/1996	LEE	Manitoba	4
	03/30/1996	GOUNOT	Ontario-South	2
	03/30/1996	GOUNOT	Quebec	18
	03/30/1996	GOUNOT	Manitoba	1
	03/31/1996	LUCCHESSI	Manitoba	1
	03/31/1996	LEE	Ontario-South	14
	03/31/1996	LEE	Ontario-North	3
	03/31/1996	LEE	Quebec	7
	03/31/1996	LEE	Manitoba	3
	03/31/1996	GOUNOT	Ontario-South	2
	03/31/1996	GOUNOT	Quebec	1
	04/01/1996	LUCCHESSI	Ontario-South	3
	04/01/1996	LUCCHESSI	Manitoba	1

Tabeller i SAMPLE-databasen

Navn:	SALES_DATE	SALES_PERSON	REGION	SALES
	04/01/1996	LEE	Ontario-South	8
	04/01/1996	LEE	Ontario-North	-
	04/01/1996	LEE	Quebec	8
	04/01/1996	LEE	Manitoba	9
	04/01/1996	GOUNOT	Ontario-South	3
	04/01/1996	GOUNOT	Ontario-North	1
	04/01/1996	GOUNOT	Quebec	3
	04/01/1996	GOUNOT	Manitoba	7

Tabellen STAFF

Navn:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
Type:	smallint not null	varchar(9)	smallint	char(5)	smallint	dec(7,2)	dec(7,2)
Beskrivelse:	Personale-nummer	Medarbejder-navn	Afdelings-nummer	Jobtype	Anciennitet	Nuværende løn	Kommission
Værdier:	10	Sanders	20	Mgr	7	18357.50	-
	20	Pernal	20	Sales	8	18171.25	612.45
	30	Marenghi	38	Mgr	5	17506.75	-
	40	O'Brien	38	Sales	6	18006.00	846.55
	50	Hanes	15	Mgr	10	20659.80	-
	60	Quigley	38	Sales	-	16808.30	650.25
	70	Rothman	15	Sales	7	16502.83	1152.00
	80	James	20	Clerk	-	13504.60	128.20
	90	Koonitz	42	Sales	6	18001.75	1386.70
	100	Plotz	42	Mgr	7	18352.80	-
	110	Ngan	15	Clerk	5	12508.20	206.60
	120	Naughton	38	Clerk	-	12954.75	180.00
	130	Yamaguchi	42	Clerk	6	10505.90	75.60
	140	Fraye	51	Mgr	6	21150.00	-
	150	Williams	51	Sales	6	19456.50	637.65
	160	Molinare	10	Mgr	7	22959.20	-
	170	Kermisch	15	Clerk	4	12258.50	110.10
	180	Abrahams	38	Clerk	3	12009.75	236.50
	190	Sneider	20	Clerk	8	14252.75	126.50
	200	Scoutten	42	Clerk	-	11508.60	84.20
	210	Lu	10	Mgr	10	20010.00	-
	220	Smith	51	Sales	7	17654.50	992.80
	230	Lundquist	51	Clerk	3	13369.80	189.65

Navn:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
	240	Daniels	10	Mgr	5	19260.25	-
	250	Wheeler	51	Clerk	6	14460.00	513.30
	260	Jones	10	Mgr	12	21234.00	-
	270	Lea	66	Mgr	9	18555.50	-
	280	Wilson	66	Sales	9	18674.50	811.50
	290	Quill	84	Mgr	10	19818.00	-
	300	Davis	84	Sales	5	15454.50	806.10
	310	Graham	66	Sales	13	21000.00	200.30
	320	Gonzales	66	Sales	4	16858.20	844.00
	330	Burke	66	Clerk	1	10988.00	55.50
	340	Edwards	84	Sales	7	17844.00	1285.00
	350	Gafney	84	Clerk	5	13030.50	188.00

Tabellen STAFFG

Bemærk: STAFFG oprettes kun, når der bruges en dobbeltbytetegntabel.

Navn:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
Type:	smallint not null	vargraphic(9)	smallint	graphic(5)	smallint	dec(9,0)	dec(9,0)
Beskrivelse:	Personale-nummer	Medarbejder-navn	Afdelings-nummer	Jobtype	Anciennitet	Nuværende løn	Kommission
Værdier:	10	Sanders	20	Mgr	7	18357.50	-
	20	Pernal	20	Sales	8	18171.25	612.45
	30	Marenghi	38	Mgr	5	17506.75	-
	40	O'Brien	38	Sales	6	18006.00	846.55
	50	Hanes	15	Mgr	10	20659.80	-
	60	Quigley	38	Sales	-	16808.30	650.25
	70	Rothman	15	Sales	7	16502.83	1152.00
	80	James	20	Clerk	-	13504.60	128.20
	90	Koonitz	42	Sales	6	18001.75	1386.70
	100	Plotz	42	Mgr	7	18352.80	-
	110	Ngan	15	Clerk	5	12508.20	206.60
	120	Naughton	38	Clerk	-	12954.75	180.00
	130	Yamaguchi	42	Clerk	6	10505.90	75.60
	140	Fraye	51	Mgr	6	21150.00	-
	150	Williams	51	Sales	6	19456.50	637.65
	160	Molinare	10	Mgr	7	22959.20	-
	170	Kermisch	15	Clerk	4	12258.50	110.10

Tabeller i SAMPLE-databasen

Navn:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
	180	Abrahams	38	Clerk	3	12009.75	236.50
	190	Sneider	20	Clerk	8	14252.75	126.50
	200	Scoutten	42	Clerk	-	11508.60	84.20
	210	Lu	10	Mgr	10	20010.00	-
	220	Smith	51	Sales	7	17654.50	992.80
	230	Lundquist	51	Clerk	3	13369.80	189.65
	240	Daniels	10	Mgr	5	19260.25	-
	250	Wheeler	51	Clerk	6	14460.00	513.30
	260	Jones	10	Mgr	12	21234.00	-
	270	Lea	66	Mgr	9	18555.50	-
	280	Wilson	66	Sales	9	18674.50	811.50
	290	Quill	84	Mgr	10	19818.00	-
	300	Davis	84	Sales	5	15454.50	806.10
	310	Graham	66	Sales	13	21000.00	200.30
	320	Gonzales	66	Sales	4	16858.20	844.00
	330	Burke	66	Clerk	1	10988.00	55.50
	340	Edwards	84	Sales	7	17844.00	1285.00
	350	Gafney	84	Clerk	5	13030.50	188.00

Eksempelfiler med BLOB- og CLOB-data

I dette afsnit vises data, som findes i filerne EMP_PHOTO (billeder af medarbejdere) og EMP_RESUME (medarbejderes CV (Curriculum Vitae)).

Billede af Quintana

Figur 7. Delores M. Quintana

CV for Quintana

Denne tekst findes i filerne db200130.asc og db200130.scr.

Resume: Delores M. Quintana

Personal Information	
Address:	1150 Eglinton Ave Mellonville, Idaho 83725
Phone:	(208) 555-9933
Birthdate:	September 15, 1925
Sex:	Female
Marital Status:	Married
Height:	5′2″
Weight:	120 lbs.
Department Information	
Employee Number:	000130
Dept Number:	C01
Manager:	Sally Kwan
Position:	Analyst
Phone:	(208) 555-4578
Hire Date:	1971-07-28
Education	
1965	Math and English, B.A. Adelphi University
1960	Dental Technician Florida Institute of Technology
Work History	
10/91 - present	Advisory Systems Analyst Producing documentation tools for engineering department.
12/85 - 9/91	Technical Writer, text programmer, and planner.
1/79 - 11/85	COBOL Payroll Programmer Writing payroll programs for a diesel fuel company.

Interests

- Cooking
- Reading
- Sewing
- Remodeling

Billede af Nicholls

Figur 8. Heather A. Nicholls

CV for Nicholls

Denne tekst findes i filerne db200140.asc og db200140.scr.

Resume: Heather A. Nicholls

Personal Information	
Address:	844 Don Mills Ave Mellonville, Idaho 83734
Phone:	(208) 555-2310
Birthdate:	January 19, 1946
Sex:	Female
Marital Status:	Single
Height:	5′8″
Weight:	130 lbs.
Department Information	
Employee Number:	000140
Dept Number:	C01
Manager:	Sally Kwan
Position:	Analyst
Phone:	(208) 555-1793
Hire Date:	1976-12-15
Education	
1972	Computer Engineering, Ph.D. University of Washington
1969	Music and Physics, M.A. Vassar College

Work History

2/83 - present	Architect, OCR Development Designing the architecture of OCR products.
12/76 - 1/83	Text Programmer Optical character recognition (OCR) programming in PL/I.
9/72 - 11/76	Punch Card Quality Analyst Checking punch cards met quality specifications.

Interests
- Model railroading
- Interior decorating
- Embroidery
- Knitting

Billede af Adamson

Figur 9. Bruce Adamson

CV for Adamson

Denne tekst findes i filerne db200150.asc og db200150.scr.

Resume: Bruce Adamson

Personal Information

Address:	3600 Steeles Ave Mellonville, Idaho 83757
Phone:	(208) 555-4489
Birthdate:	May 17, 1947
Sex:	Male
Marital Status:	Married
Height:	6′0″
Weight:	175 lbs.

Department Information

Employee Number:	000150
Dept Number:	D11
Manager:	Irving Stern
Position:	Designer
Phone:	(208) 555-4510
Hire Date:	1972-02-12

Education

1971	Environmental Engineering, M.Sc. Johns Hopkins University
1968	American History, B.A. Northwestern University

Work History

8/79 - present	Neural Network Design Developing neural networks for machine intelligence products.
2/72 - 7/79	Robot Vision Development Developing rule-based systems to emulate sight.
9/71 - 1/72	Numerical Integration Specialist Helping bank systems communicate with each other.

Interests

- Racing motorcycles
- Building loudspeakers
- Assembling personal computers
- Sketching

Billede af Walker

Figur 10. James H. Walker

CV for Walker

Denne tekst findes i filerne db200190.asc og db200190.scr.

Resume: James H. Walker

Personal Information	
Address:	3500 Steeles Ave Mellonville, Idaho 83757
Phone:	(208) 555-7325
Birthdate:	June 25, 1952
Sex:	Male
Marital Status:	Single
Height:	5′11″
Weight:	166 lbs.
Department Information	
Employee Number:	000190
Dept Number:	D11
Manager:	Irving Stern
Position:	Designer
Phone:	(208) 555-2986
Hire Date:	1974-07-26
Education	
1974	Computer Studies, B.Sc. University of Massachusetts
1972	Linguistic Anthropology, B.A. University of Toronto
Work History	
6/87 - present	Microcode Design Optimizing algorithms for mathematical functions.
4/77 - 5/87	Printer Technical Support Installing and supporting laser printers.
9/74 - 3/77	Maintenance Programming Patching assembly language compiler for mainframes.

Interests

- Wine tasting
- Skiing
- Swimming
- Dancing

Tillæg B. DB2-dokumentation

Oplysningerne om DB2 Universal Database består af onlinehjælp, bøger (PDF og HTML) og eksempelprogrammer i HTML-format. I dette afsnit beskrives oplysningerne, og hvordan du får adgang til dem.

Brug Informationscenter til at få adgang til onlineproduktinformation. Der er flere oplysninger i "Adgang til bøger vha. Informationscenter" på side 105. Du kan få vist oplysninger om opgaver, DB2-bøger, fejlfinding, eksempelprogrammer og DB2-information på WWW.

DB2 PDF-filer og trykte bøger

DB2-bøger

I følgende oversigt er DB2-bøgerne inddelt i fire kategorier:

Vejledninger og opslagsbøger til DB2
Disse bøger indeholder DB2-oplysninger, som er fælles for alle platforme.

Oplysninger om installation og konfiguration af DB2
Disse bøger gælder for en bestemt platform, hvor DB2 er installeret. Der er f.eks. forskellige brugervejledninger (*Quick Beginnings*-bøger) til DB2 under OS/2, under Windows og på UNIX-baserede platforme.

Fælles HTML-programeksempler
Eksemplerne er HTML-versionen af de programeksempler, der installeres sammen med komponenten Applikationsudviklingsklient. De er til orientering og erstatter ikke de egentlige programmer.

Versionsnoter
Her finder du de nyeste oplysninger, som ikke er med i DB2-bøgerne.

Installationsbøger, versionsnoter og øvelser kan ses i HTML direkte fra program-cd'en. De fleste bøger findes i HTML på program-cd'en og i PDF-format (Adobe Acrobat) på DB2-cd'en med bøger, hvorfra de kan fremvises og udskrives. Du kan også bestille en trykt udgave hos IBM. Se "Bestil trykte bøger" på side 101. Nedenstående oversigt viser de bøger, der kan bestilles.

På OS/2- og Windows-platforme kan HTML-filerne installeres i biblioteket sqllib\doc\html. En del af DB2-dokumentationen er oversat til andre sprog. Når oplysningerne ikke findes på et bestemt sprog, leveres de på engelsk.

På UNIX-platforme kan du installere flere sprogversioner af HTML-filerne under bibliotekerne doc/%L/html, hvor %L står for sproget. Der er flere oplysninger i den relevante brugervejledning (*Quick Beginnings*).

Der er flere måder at få adgang til DB2-bøger og få vist oplysningerne i dem:

- "Vis onlineoplysninger" på side 104
- "Søg i onlineoplysninger" på side 109
- "Bestil trykte bøger" på side 101
- "Udskriv PDF-bøger" på side 100

Tabel 1. DB2-bøger

Navn	Beskrivelse	Formnummer PDF-filnavn	HTML-bibliotek
	Vejledninger og opslagsbøger til DB2		
Administration Guide	*Administration Guide: Planning* indeholder en oversigt over databasebegreber, oplysninger om designaspekter, f.eks. logisk og fysisk databasedesign, og om høj tilgængelighed.	SC09-2946 db2d1x70	db2d0
	Administration Guide: Implementation indeholder oplysninger om implementering, f.eks. implementering af design, adgang til databaser, kontrol, sikkerhedskopiering og retablering.	SC09-2944 db2d2x70	
	Administration Guide: Performance indeholder oplysninger om databasemiljø og evaluering og tuning af applikationsperformance.	SC09-2945 db2d3x70	
	Alle tre bind af *Administration Guide* kan bestilles på engelsk vha. formnummeret SBOF-8934.		
Administrative API Reference	DB2-API'er (Application Programming Interface) og datastrukturer til styring af databaserne. Bogen beskriver også, hvordan API'er kaldes fra applikationerne.	SC09-2947 db2b0x70	db2b0
Application Building Guide	Oplysninger om konfiguration af miljøet og en trinvis vejledning i kompilering, linkning og udførelse af DB2-applikationer på Windows-, OS/2- og UNIX-baserede platforme.	SC09-2948 db2axx70	db2ax

Tabel 1. DB2-bøger (fortsat)

Navn	Beskrivelse	Formnummer PDF-filnavn	HTML-bibliotek
APPC, CPI-C, and SNA Sense Codes	Indeholder generelle oplysninger om APPC-, CPI-C- og SNA-registreringskoder, der kan forekomme under anvendelsen af DB2 Universal Database-programmer. Findes kun i HTML-format.	Intet formnummer db2apx70	db2ap
Application Development Guide	Forklaring på udvikling af applikationer, der opretter adgang til DB2-databaser vha. indlejret SQL eller Java (JDBC and SQLJ). Det forklares også, hvordan man skriver lagrede procedurer, brugerdefinerede funktioner, opretter brugerdefinerede typer, hvordan man anvender triggere, og hvordan man udvikler applikationer i inddelte miljøer eller i fødererede systemer.	SC09-2949 db2a0x70	db2a0
CLI Guide and Reference	Udvikling af applikationer, der får adgang til DB2-databaser vha. DB2-CLI (Call Level Interface), en SQL-grænseflade, som er kompatibel med Microsofts ODBC.	SC09-2950 db2l0x70	db2l0
Command Reference	Brugen af DB2-kommandolinien og DB2-kommandoer til styring af databasen.	SC09-2951 db2n0x70	db2n0
Connectivity Supplement	Konfigurations- og referenceoplysninger om anvendelsen af DB2 til AS/400, DB2 til OS/390, DB2 til MVS eller DB2 til VM som DRDA-applikations-requestere sammen med DB2 Universal Database-servere. Bogen beskriver også anvendelsen af DRDA-applikationsservere sammen med DB2 Connect-applikations-requestere. Findes kun i HTML og PDF.	Intet formnummer db2h1x70	db2h1
Data Movement Utilities Guide and Reference	Forklaring på, hvordan man anvender DB2-funktioner, f.eks. IMPORT, EXPORT, LOAD, AutoLoader og DPROP, til at flytte data.	SC09-2955 db2dmx70	db2dm

Tabel 1. DB2-bøger (fortsat)

Navn	Beskrivelse	Formnummer PDF-filnavn	HTML-bibliotek
Data Warehouse Center Administration Guide	Oplysning om opbygning og vedligeholdelse af et datavarehus vha. datavarehuscentret.	SC26-9993 db2ddx70	db2dd
Data Warehouse Center Application Integration Guide	Oplysninger til programmører om integration af applikationer med datavarehuscentret og Information Catalog Manager.	SC26-9994 db2adx70	db2ad
DB2 Connect Brugervejledning	Begreber, programmering og generel brug af DB2 Connect-programmer.	S511-5802 db2c0x70	db2c0
DB2 Query Patroller Administration Guide	Giver et driftsmæssigt overblik over DB2 Query Patroller, specifikke oplysninger om drift og administration og oplysninger om brug af funktionerne i den grafiske brugergrænseflade til administration.	SC09-2958 db2dwx70	db2dw
DB2 Query Patroller User's Guide	Beskrivelse af, hvordan man bruger værktøjer og funktioner i DB2 Query Patroller.	SC09-2960 db2wwx70	db2ww
Ordliste	Definitioner af begreber, der bruges i DB2 og DB2-komponenterne. Findes på dansk i HTML-format og på engelsk i *SQL Reference*.	Intet formnummer db2t0x70	db2t0
Image, Audio, and Video Extenders Administration and Programming	Generelle oplysninger om DB2-udvidelsesprogrammer (Extenders), administration og konfiguration af IAV Extenders (udvidelser til billeder, lyd og video) og programmering vha. IAV Extenders. Bogen indeholder også opslagsmateriale, fejlfindingsoplysninger med meddelelser og eksempler.	SC26-9929 dmbu7x70	dmbu7
Information Catalog Manager Administration Guide	Vejledning i håndtering af informationskataloger.	SC26-9995 db2dix70	db2di
Information Catalog Manager Programming Guide and Reference	Definitioner af grænsefladerne i Information Catalog Manager.	SC26-9997 db2bix70	db2bi
Information Catalog Manager User's Guide	Beskriver brugergrænsefladen i Information Catalog Manager.	SC26-9996 db2aix70	db2ai

Tabel 1. DB2-bøger (fortsat)

Navn	Beskrivelse	Formnummer PDF-filnavn	HTML-bibliotek
Installation og konfiguration	En vejledning i planlægning, installation og konfiguration af platformspecifikke DB2-klienter. Bogen indeholder også oplysninger om binding, konfiguration af client/serverkommunikation, grafiske DB2-værktøjer, DRDA-applikationsservere, distribueret installation, konfiguration af distribuerede forespørgsler og adgang til heterogene datakilder.	G511-5796 db2iyx70	db2iy
Meddelelseshåndbog	Indeholder meddelelser og koder, der afsendes af DB2, Information Catalog Manager og Datavarehuscenter og evt. handlinger i forbindelse hermed. Begge bind af Meddelelseshåndbog kan bestilles på engelsk i USA vha. formnummer SBOF-8932.	Bind 1 G511-5800 db2m1x70 Bind 2 G511-5801 db2m2x70	db2m0
OLAP Integration Server Administration Guide	Forklarer brugen af Administration Manager-komponenten i OLAP Integration Server.	SC27-0782 db2dpx70	Ikke tilgængelig
OLAP Integration Server Metaoutline User's Guide	Forklarer, hvordan man opretter og indsætter data i OLAP-metastrukturer vha. standardgrænsefladen for OLAP-metastrukturer (ikke vha. Metaoutline Assistant).	SC27-0784 db2upx70	Ikke tilgængelig
OLAP Integration Server Model User's Guide	Forklarer, hvordan man opretter OLAP-modeller vha. standardgrænsefladen for OLAP-modeller (ikke vha. Model Assistant).	SC27-0783 db2lpx70	Ikke tilgængelig
OLAP Installations- og brugervejledning	Konfiguration og installation af OLAP Starter Kit.	S511-5805 db2ipx70	db2ip
OLAP Spreadsheet Add-in Brugervejledning til Excel	Beskriver, hvordan man bruger et Excel-regneark til at analysere OLAP-data.	S511-5806 db2epx70	db2ep
OLAP Spreadsheet Add-in Brugervejledning til Lotus 1-2-3	Beskriver, hvordan man bruger et Lotus 1-2-3-regneark til at analysere OLAP-data.	S511-5807 db2tpx70	db2tp

Tabel 1. DB2-bøger (fortsat)

Navn	Beskrivelse	Formnummer PDF-filnavn	HTML-bibliotek
Replication Guide and Reference	Planlægning, konfiguration, administration og brug af IBM-replikeringsværktøjer, som leveres med DB2.	SC26-9920 db2e0x70	db2e0
Spatial Extender User's Guide and Reference	Indeholder oplysninger om, hvordan man installerer, konfigurerer, administrerer, programmerer og udfører fejlsøgning i Spatial Extender. Beskriver også begrebet rumlige data og indeholder meddelelser og SQL, der vedrører Spatial Extender.	SC27-0701 db2sbx70	db2sb
Kom godt i gang med SQL	SQL-begreber og eksempler på konstruktioner og funktioner.	S511-5803 db2y0x70	db2y0
SQL Reference, Volume 1 og Volume 2	SQL-syntaks og -semantik samt sproglige regler. Der er også oplysninger om manglende kompatibilitet mellem versioner, programbegrænsninger og katalogudpluk. Begge bind af *SQL Reference* kan bestilles på engelsk i USA vha. formnummer SBOF-8933.	SC09-2974 Volume 1 db2s1x70 SC09-2975 Volume 2 db2s2x70	db2s0
System Monitor Guide and Reference	Beskriver, hvordan forskellige typer oplysninger indsamles om databaser og databasesystemet. Indeholder en forklaring på, hvordan du bruger oplysningerne til at få en forståelse af databaseaktiviteter, forbedre performance og finde årsagen til fejl.	SC09-2956 db2f0x70	db2f0
Text Extender Administration and Programming	Generelle oplysninger om DB2-udvidelsesprogrammer (Extenders), administration og konfiguration af Text Extender og programmering vha. Text Extender. Bogen indeholder også opslagsmateriale, fejlfindingsoplysninger med meddelelser og eksempler.	SC26-9930 desu9x70	desu9
Troubleshooting Guide	Årsagen til fejl, retablering efter fejl og brug af fejlfindingsværktøjer samt DB2-service.	GC09-2850 db2p0x70	db2p0

Tabel 1. DB2-bøger (fortsat)

Navn	Beskrivelse	Formnummer PDF-filnavn	HTML-bibliotek
Nye funktioner i DB2	Beskrivelse af nye faciliteter, funktioner og forbedringer i DB2 Universal Database version 7.	S511-5804 db2q0x70	db2q0
Oplysninger om installation og konfiguration af DB2			
DB2 Connect Enterprise Edition for OS/2 and Windows Quick Beginnings	Planlægning, overførsel, installation og konfiguration af DB2 Connect Enterprise Edition under OS/2 og Windows 32-bit-styresystemer. Der er også oplysninger om installation og konfiguration af mange understøttede klienter.	GC09-2953 db2c6x70	db2c6
DB2 Connect Enterprise Edition for UNIX Quick Beginnings	Planlægning, overførsel, installation, konfiguration og brug af DB2 Connect Enterprise Edition på UNIX-baserede platforme. Der er også oplysninger om installation og konfiguration af mange understøttede klienter.	GC09-2952 db2cyx70	db2cy
DB2 Connect Personal Edition Kom godt i gang	Planlægning, overførsel, installation, konfiguration og brug af DB2 Connect Personal Edition under OS/2 og Windows 32-bit-styresystemer. Der er også oplysninger om installation og konfiguration af klienter.	G511-5797 db2c1x70	db2c1
DB2 Connect Personal Edition Quick Beginnings for Linux	Planlægning, installation, overførsel og konfiguration af DB2 Connect Personal Edition til alle understøttede Linux-distributioner.	GC09-2962 db2c4x70	db2c4
DB2 Data Links Manager Quick Beginnings	Planlægning, installation, konfiguration og opgaver i DB2 Data Links Manager til AIX og Windows 32-bit-styresystemer.	GC09-2966 db2z6x70	db2z6
DB2 Enterprise - Extended Edition for UNIX Quick Beginnings	Planlægning, installation og konfiguration af DB2 Enterprise - Extended Edition på UNIX-baserede platforme. Der er også oplysninger om installation og konfiguration af mange understøttede klienter.	GC09-2964 db2v3x70	db2v3
DB2 Enterprise - Extended Edition for Windows Quick Beginnings	Planlægning, installation og konfiguration af DB2 Enterprise - Extended Edition under Windows 32-bit-styresystemer. Der er også oplysninger om installation og konfiguration af mange understøttede klienter.	GC09-2963 db2v6x70	db2v6

Tabel 1. DB2-bøger (fortsat)

Navn	Beskrivelse	Formnummer PDF-filnavn	HTML-bibliotek
DB2 til OS/2 Brugervejledning	Planlægning, installation, overførsel og konfiguration af DB2 Universal Database til OS/2. Der er også oplysninger om installation og konfiguration af mange understøttede klienter.	G511-5798 db2i2x70	db2i2
DB2 for UNIX Quick Beginnings	Planlægning, installation, overførsel og konfiguration af DB2 Universal Database på UNIX-baserede platforme. Der er også oplysninger om installation og konfiguration af mange understøttede klienter.	GC09-2970 db2ixx70	db2ix
DB2 til Windows Brugervejledning	Planlægning, installation, overførsel og konfiguration af DB2 Universal Database under Windows 32-bit-styresystemer. Der er også oplysninger om installation og konfiguration af mange understøttede klienter.	G511-5799 db2i6x70	db2i6
DB2 Personal Edition Quick Beginnings	Planlægning, installation, overførsel og konfiguration af DB2 Universal Database Personal Edition under OS/2 og Windows 32-bit-styresystemer.	GC09-2969 db2i1x70	db2i1
DB2 Personal Edition Quick Beginnings for Linux	Planlægning, installation, overførsel og konfiguration af DB2 Universal Personal Personal Edition til alle understøttede Linux-distributioner.	GC09-2972 db2i4x70	db2i4
DB2 Query Patroller Installation Guide	Installationsoplysninger om DB2 Query Patroller.	GC09-2959 db2iwx70	db2iw
DB2 Warehouse Manager Installation Guide	Installationsoplysninger om varehusagenter, varehustransformeringer og Information Catalog Manager.	GC26-9998 db2idx70	db2id

Tabel 1. DB2-bøger (fortsat)

Navn	Beskrivelse	Formnummer PDF-filnavn	HTML-bibliotek
	Fælles HTML-programeksempler		
HTML-programeksempler	Indeholder programeksempler i HTML-format til programmeringssprog på alle platforme, som understøttes af DB2. Eksemplerne er orienterende. Ikke alle eksempler er tilgængelige i alle programmeringssprog. HTML-eksemplerne er kun tilgængelige, når DB2-applikationsudviklingsklient er installeret. Der er flere oplysninger om programmerne i *Application Building Guide*.	Intet formnummer	db2hs
	Versionsnoter		
Versionsnoter til DB2 Connect	De nyeste oplysninger, som ikke er med i DB2 Connect-bøgerne.	Se bemærkning 2.	db2cr
Installationsnoter til DB2	De nyeste installationsoplysninger, som ikke er med i DB2-bøgerne.	Findes kun på program-cd'en.	
Versionsnoter til DB2	De nyeste oplysninger om alle DB2-programmer og -faciliteter, som ikke er med i DB2-bøgerne.	Se bemærkning 2.	db2ir

Bemærkninger:

1. Det sjette tegn, *x*, i filnavnet angiver bogens sprogudgave. Filnavnet `db2c0e70` angiver f.eks. den engelske udgave af *DB2 Connect Brugervejledning*, og filnavnet `db2c0d70` angiver den danske udgave af samme bog. Der er brugt følgende bogstaver i sjette position i filnavnet til at angive sprogudgaven:

Sprog	Id
Brasiliansk portugisisk	b
Bulgarsk	u
Dansk	d
Engelsk	e
Finsk	y
Forkortet kinesisk	c
Fransk	f
Græsk	a
Hollandsk	q
Italiensk	i
Japansk	j

Koreansk	k
Norsk	n
Polsk	p
Portugisisk	v
Russisk	r
Slovensk	l
Spansk	z
Svensk	s
Tjekkisk	x
Tyrkisk	m
Tysk	g
Uforkortet kinesisk	t
Ungarsk	h

2. I versionsnoterne findes de nyeste oplysninger, som ikke er med i DB2-bøgerne. De findes i HTML-format og som en ASCII-fil. HTML-udgaven er tilgængelig fra Informationscenter og på program-cd'erne. Sådan får du vist ASCII-filen:
   - På UNIX-baserede platforme skal du se i filen Release.Notes. Filen er placeret i biblioteket DB2DIR/Readme/%*L*, hvor *%L* er navnet på sprog-konventionerne, og `DB2DIR` er:
     - /usr/lpp/db2_07_01 i AIX
     - /opt/IBMdb2/V7.1 i HP-UX, PTX, Solaris og Silicon Graphics IRIX
     - /usr/IBMdb2/V7.1 i Linux.
   - På andre platforme skal du se i filen RELEASE.TXT. Filen er placeret i det bibliotek, hvor programmet er installeret. Under OS/2 kan du også åbne folderen **IBM DB2** og dobbeltklikke på ikonen **Versionsnoter**.

Udskriv PDF-bøger

Hvis du foretrækker at udskrive bøgerne, kan du udskrive de PDF-filer, der findes på DB2-cd'en med bøger. Vha. Adobe Acrobat Reader kan du enten udskrive hele bogen eller et bestemt sideinterval. Filnavnet på de enkelte bøger i biblioteket findes i tabel 1 på side 92.

Du kan hente den seneste version af Adobe Acrobat Reader fra Adobe's Web-side http://www.adobe.com.

PDF-filerne findes på DB2-cd'en med bøger. De har filtypen PDF. Sådan får du adgang til PDF-filerne:

1. Sæt DB2-cd'en med bøger i drevet. På UNIX-baserede platforme skal du tilknytte cd'en. Tilknytning af cd-drevet er beskrevet i *Quick Beginnings*.
2. Start Acrobat Reader.
3. Åbn den pågældende PDF-fil fra et af følgende steder:
   - OS/2 og Windows:

Biblioteket *x*:\doc*sprog*, hvor *x* repræsenterer cd-drevet, og *sprog* er den landekode på to bogstaver, der repræsenterer sproget, f.eks. DK for dansk.

- UNIX-baserede platforme:

  Biblioteket */cdrom*/doc/*%L* på cd'en, hvor */cdrom* repræsenterer cd'ens tilknytningspunkt, og *%L* står for sproget.

Du kan også kopiere PDF-filerne fra cd'en til et lokalt drev eller et netværksdrev og læse dem derfra.

Bestil trykte bøger

Du kan bestille trykte DB2-bøger enkeltvis eller som et sæt. Bogsæt kan kun bestilles på engelsk i USA vha. et SBOF-nummer (Sold Bill of Forms). Bøger kan bestilles hos en IBM-forhandler. Du kan også bestille bøger på Web-siden for IBM-bøger på http://www.elink.ibmlink.ibm.com/pbl/pbl.

Der findes to bogsæt. SBOF-8935 indeholder opslagsmateriale og oplysninger om brug til DB2 Warehouse Manager. SBOF-8931 indeholder opslagsmateriale og oplysninger om brug til alle andre DB2 Universal Database-produkter og -faciliteter. Indholdet af hvert bogsæt er vist i nedenstående tabel:

Tabel 2. Bestil trykte bøger

SBOF-nummer	Indeholdte bøger	
SBOF-8931	• Administration Guide: Planning • Administration Guide: Implementation • Administration Guide: Performance • Administrative API Reference • Application Building Guide • Application Development Guide • CLI Guide and Reference • Command Reference • Data Movement Utilities Guide and Reference • Data Warehouse Center Administration Guide • Data Warehouse Center Application Integration Guide • DB2 Connect User's Guide • Installation and Configuration Supplement • Image, Audio, and Video Extenders Administration and Programming • Message Reference, Volume 1 og 2	• OLAP Integration Server Administration Guide • OLAP Integration Server Metaoutline User's Guide • OLAP Integration Server Model User's Guide • OLAP Integration Server User's Guide • OLAP Setup and User's Guide • OLAP Spreadsheet Add-in User's Guide for Excel • OLAP Spreadsheet Add-in User's Guide for Lotus 1-2-3 • Replication Guide and Reference • Spatial Extender Administration and Programming Guide • SQL Getting Started • SQL Reference, Volume 1 og 2 • System Monitor Guide and Reference • Text Extender Administration and Programming • Troubleshooting Guide • What's New
SBOF-8935	• Information Catalog Manager Administration Guide • Information Catalog Manager User's Guide • Information Catalog Manager Programming Guide and Reference	• Query Patroller Administration Guide • Query Patroller User's Guide

DB2-onlinedokumentation

Onlinehjælp

Der findes onlinehjælp til alle DB2-komponenter. Følgende oversigt beskriver de forskellige typer hjælp.

Hjælpetype	Indhold	Adgang
Hjælp til kommandoer	Forklaring på syntaks i kommandoer på DB2-kommandolinien.	Fra DB2-kommandolinien i interaktiv tilstand skal du skrive: `? kommando` hvor *kommando* er et nøgleord eller hele kommandoen. Eksempel: Hvis du skriver `? catalog` får du vist hjælp til alle CATALOG-kommandoer, og ved at skrive `? catalog database` får du vist hjælp til kommandoen CATALOG DATABASE.
Klientkonfiguration - hjælp **Kommandocentral - hjælp** **Kontrolcenter - hjælp** **Datavarehuscenter - hjælp** **Aktivitetsanalyse - hjælp** **Information Catalog Manager - hjælp** **Satellitadministration - hjælp** **Kommandofiler - hjælp**	Forklaring på opgaver, du kan udføre i et vindue eller en notesbog. Hjælpen omfatter bl.a. en oversigt og grundlæggende oplysninger, du kan få brug for, og brugen af elementerne i vinduet eller notesbogen forklares.	Fra et vindue eller en notesbog skal du vælge trykknappen **Hjælp** eller trykke på **F1**.

Hjælpetype	Indhold	Adgang

Hjælp til meddelelser	Beskrivelse af årsagen til en meddelelse samt en eventuel handling.	Fra DB2-kommandolinien i interaktiv tilstand skal du skrive:
		`? XXXnnnnn`
		hvor *XXXnnnnn* er en gyldig meddelelses-id.
		Eksempel: Hvis du skriver `? SQL30081` får du vist hjælp til meddelelsen SQL30081.
		Hvis du vil have vist ét skærmbillede ad gangen i hjælpen til meddelelser, skal du skrive:
		`? XXXnnnnn | more`
		Hvis du vil gemme hjælpen til en meddelelse i en fil, skal du skrive:
		`? XXXnnnnn`
		`> filnavn.typ`
		hvor *filnavn.typ* er den fil, hvor hjælpen skal gemmes.

Hjælp til SQL	Forklaring på syntaksen i SQL-sætninger.	Fra DB2-kommandolinien i interaktiv tilstand skal du skrive: `help sætning` hvor *sætning* er en SQL-sætning. Eksempel: Hvis du skriver help SELECT, får du vist hjælp til SELECT-sætningen. **Bemærk:** Der er ingen hjælp til SQL på UNIX-baserede platforme.
Hjælp til SQLSTATE	Forklaring på SQLSTATE-værdier og klassekoder.	Fra DB2-kommandolinien i interaktiv tilstand skal du skrive: `? sqlstate eller ? klassekode` hvor *sqlstate* er en gyldig femcifret SQLSTATE-værdi, og *klassekode* er de to første cifre af SQLSTATE-værdien. Eksempel: Hvis du skriver `? 08003`, får du vsst hjælp til SQLSTATE 08003. Hvis du skriver `? 08`, får du vist hjælp til klassekode 08.

Vis onlineoplysninger

Bøgerne til programmet er i formatet HTML (Hypertext Markup Language). Det elektroniske format gør det lettere at søge og få vist oplysninger, og du kan benytte link til at få vist beslægtede oplysninger. Det er også lettere, når flere brugere er fælles om bøgerne.

Du kan få vist onlinebøger eller programeksempler med alle browsere, der overholder HTML Version 3.2-specifikationerne.

Sådan får du vist onlinebøger eller programeksempler:

- Hvis du bruger DB2-administratorværktøjer, kan du bruge informationscentret.
- Klik på **Fil —>Åbn side** i en browser. Den side, der vises, indeholder beskrivelser af og link til DB2-bøgerne:
  - På UNIX-baserede platforme skal du åbne følgende side:

    INSTHOME/sqllib/doc/*%L*/html/index.htm

    hvor *%L* er det sprog, der skal bruges.
  - På andre platforme skal du åbne følgende side:

    SQLLIB\DOC\HTML\INDEX.HTM

    Stien findes på det drev, hvor DB2 er installeret.

    Hvis du ikke har installeret Informationscenter, kan du åbne siden ved at dobbeltklikke på ikonen **DB2-onlinehjælp**. Afhængigt af systemet er ikonen placeret i DB2-folderen eller i startmenuen til Windows.

Installér Netscape-browser

Hvis du ikke har installeret en Web-browser, kan du installere Netscape fra Netscape-cd'en, der findes i programpakken. Gør følgende, hvis du vil have vist detaljerede oplysninger om, hvordan programmet installeres:

1. Indsæt Netscape-cd'en.
2. På UNIX-baserede platforme skal cd-drevet tilknyttes. Tilknytning af cd'en er beskrevet i *Quick Beginnings*.
3. Der findes en installationsvejledning i filen CDNAV*nn*.txt, hvor *nn* er sprog-id'en på to bogstaver. Filen findes i hovedbiblioteket på cd'en.

Adgang til bøger vha. Informationscenter

Gennem informationscentret får du hurtig adgang til DB2-bøgerne. Informationscentret er tilgængeligt på alle platforme, hvor DB2-administratorværktøjerne er tilgængelige.

Du kan åbne informationscentret ved at dobbeltklikke på ikonen Informationscenter. Afhængigt af systemet findes ikonen i informationsfolderen i DB2-folderen eller på **Start**-menuen i Windows.

Du kan også få adgang til informationscentret vha. værktøjslinien og menuen **Hjælp** i DB2 til Windows.

Informationscentret indeholder seks typer oplysninger. Klik på det relevante skilleblad for at se de emner, der findes til den pågældende type.

Opgaver Centrale opgaver, der kan udføres vha. DB2.

Opslag DB2-referenceoplysninger, f.eks. nøgleord, kommandoer og API'er.

Bøger DB2-bøger.

Fejlfinding Kategorier med fejlmeddelelser og de tilhørende handlinger.

Programeksempler
Programeksempler, som leveres sammen med DB2-applikationsudviklingsklient. Hvis du ikke har installeret DB2-applikationsudviklingsklient, vises skillebladet ikke.

WWW DB2-oplysninger på World Wide Web. Hvis du vil have adgang til oplysningerne, skal du have forbindelse til WWW fra systemet.

Når du vælger et punkt på en af oversigterne, startes der automatisk et fremvisningsprogram fra informationscentret, så du kan få vist oplysningerne. Fremvisningsprogrammet kan være systemets indbyggede program, et redigeringsprogram eller en Web-browser, afhængigt af de oplysninger, du vælger at få vist.

Informationscentret indeholder en søgefunktion, så du kan søge efter et bestemt emne uden at gennemgå oversigterne.

Hvis du vil foretage en fuldstændig tekstsøgning, skal du vælge knappen Søg for at få vist søgeformularen **Søg i DB2-onlinehjælp**.

HTML-søgeserveren startes som regel automatisk. Hvis en HTML-søgning ikke fungerer, skal du muligvis starte søgeserveren på en af følgende måder:

I Windows
Klik på **Start**, og vælg **Programmer —> IBM DB2 —> Start HTML-søgeserver**.

I OS/2:
Dobbeltklik på folderen **DB2 til OS/2** og derefter på ikonen **Start HTML-søgeserver**.

Se i versionsnoterne, hvis du har andre problemer med søgning i HTML-dokumenter.

Bemærk: Søgefunktionen er ikke tilgængelig i Linux-, PTX- og Silicon Graphics IRIX-miljøer.

DB2-guider

Guiderne indeholder hjælp til bestemte administrative funktioner i form af trinvise vejledninger. Guiderne er tilgængelige via Kontrolcenter og Klientkonfiguration. Nedenstående tabel indeholder en oversigt over guiderne og deres formål.

Bemærk: Guiderne Opret database, Opret indeks, Konfigurér multiopdatering og Konfigurér performance er tilgængelige i et miljø med inddelte databaser.

Guide	Hjælp til at...	Adgang
Tilføj database	Katalogisere en database på en klientarbejdsstation.	Vælg **Tilføj** fra Klientkonfiguration.
Sikkerhedskopiér database	Udarbejde, oprette og planlægge sikkerhedskopiering.	Klik i kontrolcentret med højre museknap på den database, du vil sikkerhedskopiere, og vælg **Sikkerhedskopiér —> Database vha. guide**.
Konfigurér multiopdatering	Konfigurere en multiopdatering, en distribueret transaktion eller en tofase-commit.	Klik i kontrolcentret med højre museknap på ikonen **Databaser**, og vælg **Multiopdatering**.
Opret database	Oprette en database og udføre grundlæggende konfigurationsopgaver.	Klik i kontrolcentret med højre museknap på folderen **Databaser**, og vælg **Opret —> Database vha. guide**.
Opret tabel	Vælge grundlæggende datatyper og oprette en primærnøgle til tabellen.	Klik i kontrolcentret med højre museknap på ikonen **Tabeller**, og vælg **Opret** —> **Tabel vha. guide**.
Opret tablespace	Oprette et nyt tablespace.	Klik i kontrolcentret med højre museknap på ikonen **Tablespaces**, og vælg **Opret** —> **Tabelspace vha. guide**.
Opret indeks	Få anbefalet, hvilke indekser der skal oprettes og slettes for alle dine forespørgsler.	Klik i kontrolcentret med højre museknap på folderen **Indekser**, og vælg **Opret** —> **Indeks vha. guide**.

Guide	Hjælp til at...	Adgang
Konfigurér performance	Optimere performance for en database ved at opdatere konfigurationsparametrene på basis af den typiske databaseanvendelse.	Klik i kontrolcentret med højre museknap på den database, du vil tune, og vælg **Konfigurér performance vha. guide**. I et inddelt databasemiljø skal du i oversigten over databaseafsnit klikke med højre museknap på det første databaseafsnit, du vil tune, og vælge **Konfigurér performance vha. guide**.
Genindlæs database	Retablere en database efter en fejl. Du får vejledning i, hvilken sikkerhedskopi og hvilke logfiler du skal bruge.	Klik i kontrolcentret med højre museknap på den database, du vil genindlæse, og vælg **Genindlæs —> Database vha. guide**.

Konfigurér dokumentserver

Som standard installeres DB2-bøger på det lokale system. Det betyder, at alle brugere, som skal have adgang til DB2-bøgerne, skal installere de samme filer. Du kan i stedet installere DB2-oplysningerne ét sted ved at udføre følgende trin:

1. Kopiér alle filer og underbiblioteker fra \sqllib\doc\html på det lokale system til en Web-server. Hver bog har sit eget underbibliotek med alle HTML- og GIF-filerne til bogen. Sørg for, at bibliotheksstrukturen ikke ændres.
2. Konfigurér Web-serveren til at søge efter filerne på den nye placering. Der er flere oplysninger i tillægget om NetQuestion i *Installation og konfiguration*.
3. Hvis du bruger Java-versionen af informationscentret kan du angive en basis-URL til alle HTML-filer. Du bør bruge URL'en til bogoversigten.
4. Når du kan få vist bogfilerne, kan du sætte bogmærker ved emner, som bruges ofte. Det kan f.eks. være praktisk at placere et bogmærke på følgende sider:
   - Bogoversigt
   - Indholdsfortegnelser i bøger, som bruges meget
   - Artikler, der ofte læses, f.eks. emnet Ret tabel
   - Søgeformular

Der er flere oplysninger om, hvordan du kan stille onlinedokumentationsfilerne til DB2 Universal Database til rådighed på en central maskine, i NetQuestion-tillægget i bogen *Installation og konfiguration*.

Søg i onlineoplysninger

Du kan søge efter oplysninger i HTML-filerne på en af følgende måder:

- Klik på **Søg** i øverste ramme. Brug søgeformularen til at finde et emne. Søgefunktionen er ikke tilgængelig i Linux-, PTX- og Silicon Graphics IRIX-miljøer.
- Klik på **Stikord** i øverste ramme. Du kan bruge stikordsregistret til at finde et bestemt emne i bogen.
- Aktivér indholdsfortegnelsen eller stikordsregistret til hjælpen eller HTML-bogen, og brug derefter Web-browserens søgefunktion til at finde et bestemt emne i bogen.
- Du kan bruge Web-browserens bogmærkefunktion til hurtigt at vende tilbage til et bestemt emne.
- Informationscentrets søgefunktion kan bruges til at søge efter bestemte emner. Der er flere oplysninger under "Adgang til bøger vha. Informationscenter" på side 105.

Tillæg C. Om dette dokument

Dette dokument kan indeholde henvisninger til eller oplysninger om IBM-produkter (maskiner eller programmer), -programmering eller -ydelser, som ikke er introduceret i Danmark. Sådanne henvisninger eller oplysninger betyder ikke nødvendigvis, at IBM på et senere tidspunkt vil introducere det pågældende i Danmark. Henvisning til IBM-produkter, -programmer eller -serviceydelser betyder ikke, at kun IBM-produkter, -programmer eller -serviceydelser kan benyttes.

IBM kan have patenter eller udestående patentansøgninger inden for det tekniske område, som dette dokument dækker. De opnår ikke licens til disse patenter eller patentansøgninger ved at være i besiddelse af dokumentet. Spørgsmål vedrørende licens skal stilles skriftligt til:

Director of Commercial Relations - Europe
IBM Deutchland GmbH
Schönaicher Strasse 220
D - 7030 Böblingen
Tyskland

Dokumentet kan indeholde tekniske unøjagtigheder eller trykfejl. Der foretages med mellemrum ændringer af oplysningerne i dokumentet. Disse ændringer inkorporeres i nye udgaver af dokumentet. IBM kan når som helst og uden varsel foretage forbedringer og/eller ændringer af de produkter og/eller programmer, der er beskrevet i dokumentet.

Eventuelle henvisninger til ikke-IBM Web-steder er kun ment som serviceoplysninger og er ikke udtryk for, at IBM giver sin støtte til disse Web-steder. De materialer, De finder på sådanne Web-steder, udgør ikke en del af materialerne til dette IBM-produkt, og brugen af Web-stederne sker for Deres egen risiko.

Hvis der er kommentarer til indholdet af dokumentet, bedes disse sendt til IBM Danmark A/S, der forbeholder sig ret til at benytte oplysningerne.

Brugere, som har licens til dette program og ønsker oplysninger om det med henblik på a) at udveksle oplysninger mellem uafhængigt udviklede programmer og andre programmer (herunder dette) og b) gensidig brug af udvekslede oplysninger, skal kontakte:

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East

North York, Ontario
M3C 1H7
Canada

Det licensprogram, der er beskrevet i dette dokument, og al licenseret materiale til licensprogrammet, leveres af IBM i henhold til IBM's Generelle Vilkår samt IBM's Internationale Program Licens Aftale (IPLA).

Alle data vedrørende ydeevne i dokumentet er opnået i et kontrolleret driftsmiljø. De resultater, der opnås i andre driftsmiljøer, kan afvige væsentligt fra de angivne data. Nogle af målingerne kan være foretaget på systemer på udviklingsniveau, og det er ikke sikkert, at samme resultater opnås på generelt tilgængelige systemer. Nogle måleresultater er anslået ved hjælp af ekstrapolering. De faktiske resultater kan afvige herfra. De bør derfor kontrollere de pågældende data for Deres specifikke miljø.

Oplysninger om ikke-IBM-produkter er indhentet fra leverandørerne af disse produkter, fra deres annonceringer eller fra andre offentligt tilgængelige kilder. IBM har ikke testet disse produkter og indestår ikke for nøjagtigheden af de angivne oplysninger om ydeevne, kompatibilitet eller andre påstande vedrørende ikke-IBM-produkter. Spørgsmål vedrørende ikke-IBM-produkters funktioner skal rettes til leverandørerne af de pågældende produkter.

Erklæringer vedrørende IBM's fremtidige udvikling eller planer er kun udtryk for målsætninger og kan ændres eller trækkes tilbage uden varsel.

Dokumentet kan indeholde eksempler på data og rapporter, som bruges i forbindelse med en virksomheds daglige forretningsgange.

Copyrightlicens:

Dokumentet kan indeholde eksempler på applikationsprogrammer i kildesprog, som viser programmeringsteknikker på forskellige styresystemsplatforme. De må kopiere, ændre og distribuere disse programeksempler i en hvilken som helst form, uden betaling til IBM, med det formål at udvikle, anvende, markedsføre eller distribuere applikationsprogrammer, som er i overensstemmelse med programmeringsgrænsefladen til det styresystem, som programeksemplerne er skrevet til. Disse eksempler er ikke testet fuldt ud under alle forhold. IBM kan derfor ikke stå inde for disse programeksemplers driftssikkerhed, serviceegnethed eller funktionsdygtighed.

Enhver hel eller delvis kopi af disse programeksempler eller af afledte arbejder deraf skal indeholde en copyrighterklæring svarende til følgende:

Varemærker

Følgende varemærker tilhører International Business Machines Corporation:

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

Følgende varemærker tilhører andre firmaer:

Varemærkerne Microsoft, Windows og Windows NT tilhører Microsoft Corporation.

Varemærket Java og alle Java-baserede varemærker og logoer og varemærket Solaris tilhører Sun Microsystems, Inc.

Varemærkerne Tivoli og NetView tilhører Tivoli Systems Inc.

Varemærket UNIX gives i licens gennem X/Open Company Limited.

Alle andre varemærker anerkendes.

Stikordsregister

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Y

Kontakt IBM

Hvis du har et teknisk problem, bør du gennemgå og udføre de handlinger, der foreslås i *Troubleshooting Guide*, inden du kontakter DB2 Service. Denne vejledning indeholder forslag til oplysninger, du kan indsamle, så DB2 Service bedre kan hjælpe.

Du kan få oplysninger om eller bestille DB2 Universal Database-programmer ved at kontakte en IBM-forhandler eller en IBM Business Partner.

I USA kan du ringe til et af følgende numre:

- Kundeservice: 1-800-237-5511
- Tilgængelig service oplyses på 1-888-426-4343

Produktinformation

I USA kan du ringe til et af følgende numre:

- Bestilling af produkter eller generelle oplysninger: 1-800-IBM-CALL (1-800-426-2255) eller 1-800-3IBM-OS2 (1-800-342-6672).
- Bestilling af bøger: 1-800-879-2755.

http://www.ibm.com/software/data/
DB2's WWW-sider indeholder aktuelle oplysninger om nyheder, produktbeskrivelser, uddannelsestilbud, osv.

http://www.ibm.com/software/data/db2/library/
Via DB2 Product and Service Technical Library kan du få adgang til FAQ (Frequently Asked Questions), rettelser, bøger og dagsaktuelle tekniske DB2-oplysninger.

Bemærk: Disse oplysninger er næsten udelukkende på engelsk.

http://www.elink.ibmlink.ibm.com/pbl/pbl/
Dette Web-sted til international bestilling af bøger indeholder oplysninger om bogbestilling.

http://www.ibm.com/education/certify/
Professional Certification Program fra Web-stedet indeholder oplysninger om certificeringstest for en række IBM-produkter, herunder DB2.

ftp.software.ibm.com
Du kan logge på som brugeren anonymous. I kataloget `/ps/products/db2` finder du demoer, rettelser, oplysninger og værktøjer til DB2 og mange andre produkter.

comp.databases.ibm-db2, bit.listserv.db2-l
Via disse Internetnyhedsgrupper kan brugerne diskutere deres erfaringer med DB2-produkterne.

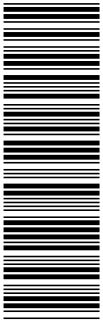
I Compuserve: GO IBMDB2
Brug kommandoen til at få adgang til forumer for IBM DB2-programmerne. Alle DB2-programmerne understøttes via disse forumer.

I tillæg A i **IBM Software Support Handbook** kan du få at vide, hvordan IBM kontaktes uden for USA. Åbn Web-siden http://www.ibm.com/support/, og vælg linket IBM Software Support Handbook nederst på siden.

Bemærk: I visse lande skal IBM-autoriserede forhandlere kontakte deres forhandlerstøttefunktion og ikke IBM eller en IBM Business Partner.

Partnummer: CT7YHDA

Printed in Denmark by IBM Danmark A/S



CT7YHDA

S511-5803-00